MASTER IN COMPUTER SCIENCE

Using goal modelling languages for representing business strategies (+ démo)

Silverio, Julien

Award date:
2014

Awarding institution:
University of Namur

Université de Namur
Faculté d'informatique
Année académique 2013–2014

Using goal modelling languages for representing business strategies

Julien SILVERIO

Maître de stage : Mme Jelena ZDRAVKOVIC et M. Constantinos GIANNOULIS

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Michaël PETIT

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Abstract

Business Strategies have been used in the domain of economics for many years now, with the goal of improving an enterprise's position in a market. They were solely used in economic domains, since they didn't translate well into an information systems (IS) world. This is the reason for the use of a goal modelling language, which by definition is used to represent the way goals for various actors are achieved. Understanding how to translate business strategy notions into goal modelling ones, but also finding adequate business strategies that could be translated into goal models and making these translations or mappings available in a modelling environment for easier use, would result in the ability of representing business strategies in IS domains and thus solving the problem mentioned above. Our work focuses on the use of the goal modelling language i* to represent three different business strategies, Strategy Maps and Balanced Scorecards, Blue Ocean Strategy and Value Configuration, as well as the use of the modelling environment ADOxx for the creation of the mapping library.

Les stratégies business sont utilisées depuis de nombreuses années dans le domaine de l'économie, avec comme but d'améliorer la position d'une entreprise dans son secteur d'acivité. Elles n'étaient utilisées que dans le domaine économique et ne se traduisaient pas bien dans le monde des systèmes d'informations. Ceci est la raison pour l'utilisation d'un langage de modélisation orienté but, qui part définition est utilisé pour représenter la façon dont les buts de différents acteurs sont réalisés. Comprendre comment traduire les notions de stratégies business en notions de modélisation orientée but, mais aussi trouver des stratégies business qui pourraient être traduites dans des modèles orientés but, ainsi que rendre ces traductions ou mappings disponibles dans un environement de modélisation pour les rendre facilement applicables, résulterait dans la capacité de représenter des stratégies business dans des systèmes d'information et donc de résoudre le problème mentioné précédemment. Notre travail se concentre sur l'utilisation du langage de modélisation orienté but, i*, pour représenter trois stratégies business différentes, Strategy Maps et Balanced Scorecards, Blue Ocean Strategy et Value Configuration, ainsi que l'environement de modélisation ADOxx pour la création d'une librairie de mappings.

Acknowledgements

The work described in this thesis is the result of a three and a half months internship at the Department of Computer and Systems Sciences of the University of Stockholm in Stockholm, Sweden. During this stay, I was under the supervision of Dr. Jelena Zdravkovic and Dr. Constantinos Giannoulis (PhD student at the time).

I would like to thank both Jelena and Constantinos for their patience and guidance throughout this whole adventure.

I would like to thank Dr. Michaël Petit and the University of Namur for giving me the opportunity of doing such an internship in one of the most beautiful cities in the world.

I would like to thank Wilfrid Utz from the University of Vienna for his help during the understanding and use of ADOxx and AdoScript.

Of course this whole experience wouldn't have been possible without the help from both my parents and I would also like to thank them for their support throughout my studies.

Last but not least, I would like to thank my girlfriend for coping with the distance between us during this internship and for her daily support and love.

Contents

List of Figures

List of Tables

Acronyms

BOS: Blue Ocean Strategy

SMBSC: Strategy Maps & Balanced Scorecards

UBSMM: Unified Business Strategy Meta-model

VC: Value Configuration

UML: Unified Modelling Language

CC: Command Call

CHAPTER 1

Introduction

1 Problem description

A business strategy is an action plan, which possesses as its main goal, the achievement of the long-term objectives of an enterprise.[2] [3] [4]
The fact that objectives are the goal of every business strategy, sparked the question of the representation of business strategies through a goal modelling language. The question is justified, since goal modelling languages are used to represent the way various goals for one or more actors are to be satisfied, which shows that business strategies and goal modelling languages are similar in regards to their uses.[2] [4] [3]
Trying to create the mappings from a business strategy to a goal modelling language is the task that will be treated in this thesis and we will show that it requires a lot more thought than one might expect from elements that seem to be so similar. [2] [3] [4]

An additional question that was raised due to the use of a goal modelling language, is in what environment these eventual mappings are to be modelled in. If we talk about mappings, we talk about constraints that need to be maintained for the model to represent what it should. The use of a particular environment and of a special library for the modelling of business strategies through goal modelling are two elements that will also be treated in this thesis. [2] [3] [**?**]

2 Structure

This thesis is structured as follows:
Part I will present a chapter about the background information concerning Business Strategy and more particularly Strategy Maps and Balanced Scorecards, Blue Ocean Strategy and Value Configuration, but also concerning the Unified Business Strategy Meta-Model, the goal modelling language i* and finally the modelling platform ADOxx. The following chapter looks at explaining the work that had already been done before the start of this thesis.

Part II of this thesis is dedicated to the theoretical contributions. We are going to explain the process behind these contributions in chapter 4 and describe the definition of the mappings for the three business strategies in chapter 5. We will finish this part with chapter 6, where we will draw first conclusions following the creation of the mapping tables.

In part III we will look at the practical contributions of this thesis. Chapter 7 will list what the goal of this part is, while chapter 8 will give us a first look on how we plan to achieve these goals. We will then take a look at the implementation and choices that had to be made in chapter 9, before drawing intermediate conclusions about the practical part of the thesis.

Part IV will be where we will present a summary of the thesis, followed by the results of our findings and contributions and a section about suggestions for future work in this domain.

Part I

Background and related work

CHAPTER 2

Background

1 Business strategy

Strategic planning is the notion that describes the process of defining a strategy, through the analysis of the current status of an enterprise as well as the competitive environment this enterprise resides in. A business strategy is the determination of both the long-term goals and the courses of action, which require the use of resources to be achieved. The organizational structure is then deduced from the examination of the two notions mentioned before, whereas the business strategy is used to define the way an organization actualizes its goals and fulfils its purposes. [2] [3] [4] [6]

Additionally, based on the reality of attaining long-term goals, there is a need to make strategies prone to varying environments, due to both internal or external developments. [2] [3] [4]
"Internal development refers to leveraging internal strengths and avoiding internal weaknesses, while external development refers to the leveraging of external opportunities and the need to foresee external threats". [6]

Three views on strategy-shaping based on the competition in microeconomics have been identified and described as being complementary to one another by Jay B. Barney in *Types of Competition and the Theory of Strategy: Toward an Integrative Framework*. This statement results in finding that strategies that consider all three types have an increased likelihood of sustainability and prosperity and thus represent a desired state for a strategy. [2] [3] [4] [6]

The three views or types are as follows:

Industrial organization

This view suggests the fact, that a competitive advantage results from the clear positioning of an enterprise in regards to its environment, which is described by the structure of the industry based on the five forces model of Porter for example.[4] [6]

Chamberlinian or resource-based

This view suggests that the competitive advantage of an enterprise depends on its distinctive capabilities provided by this enterprise's resources. [4]

Schumpeterian

This view suggests that the industrial environment in which an enterprise resides is subject to grave disturbances if unanticipated and radical innovations were to occur. These disturbances could result in opportunities given to the enterprise, to take an advantage over its competitors.[4] [6]

1.1 Strategy Maps and Balanced Scorecards

Strategy maps and balanced scorecards (SMBSC) is a framework resulting from the combination of two tools proposed by Kaplan and Norton to accurately represent, communicate and monitor business strategy as well as strategic objectives. [2] [4] [5] [6] [7] [9]

The first tool was introduced by Kaplan and Norton in 1996 and is called the scorecard. It consists of "strategic objectives and related measures, which include concrete targets and initiatives towards their achievement". Additionally the monitoring and assessment of cause-effect links, which structure the scoreboards, are essential for the identification of interdependencies throughout an organization. [4] [6] [7] [9]
A balanced scorecard provides two elements, the first is a representation through four organizational perspectives of an organization's business activities and the second allows the communication of priorities throughout an enterprise. [6]
The four perspectives - financial, customer, internal and learning & growth - allow coverage of business processes and with the addition of short-term targets and a bottom up view of the perspectives to address the time aspect, we have the first part of what makes a scorecard balanced. [6]
The second part that makes a scorecard balanced, is that it covers both internal and external aspects of an enterprise.

The second tool was introduced by Kaplan and Norton in 2001 and is called strategy map. It serves as a mediator between different elements like core values, vision and the strategy of an enterprise and the work that is performed. They also proposed a template, which would help represent the way an enterprise creates value and which positioned this template as one of the few existing templates, that allowed a visual representation of a strategy. [4] [6] [7] [9]
The guidelines for creating such a strategy map included the need to follow a top-down approach, starting with the creation of the mission statement and the core values and working ones way down to organizational elements like hiring specialized personnel. [4] [6] [7] [9]

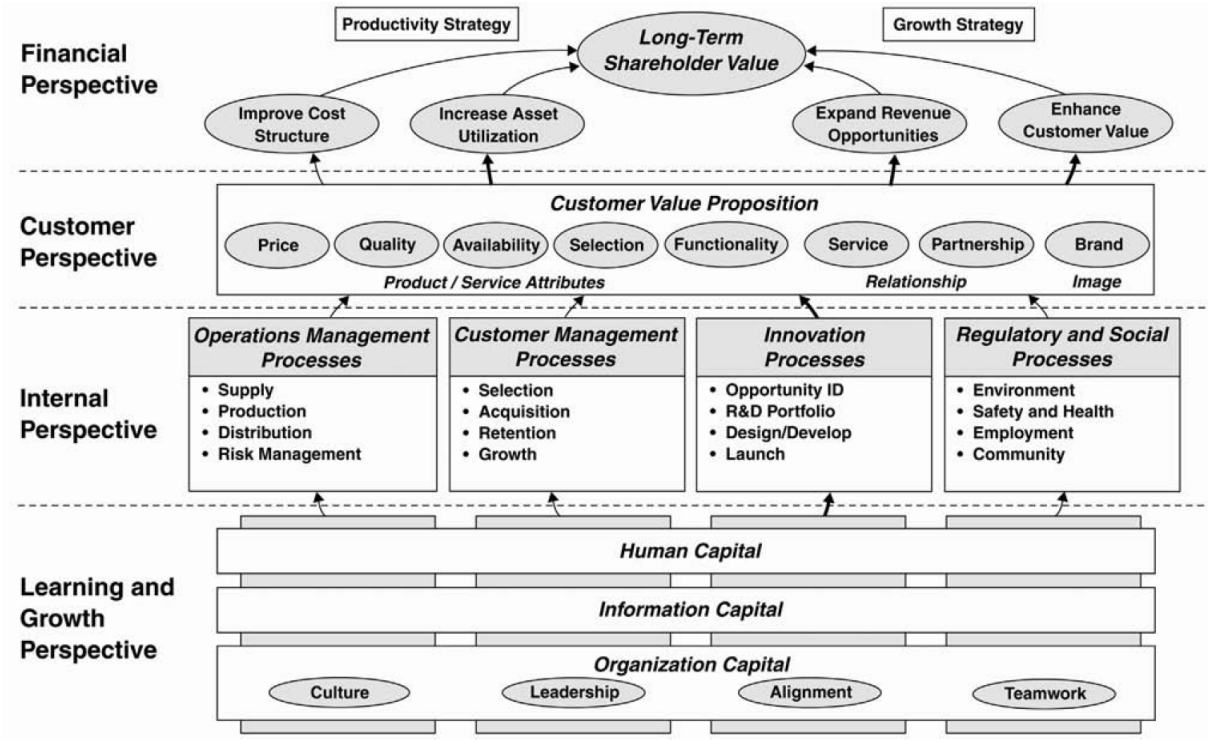

Figure 2.1: The Strategy Map template [10]

Additionally a strategy map is a general representation of the four organizational perspectives of the balanced scorecard and according to Kaplan and Norton is based on five principles: [4] [6]

- A strategy needs to balance long-term financial commitments with the goal of increasing profits, but also short-term commitments with the goal of reducing cost and improving productivity. [6] (Financial perspective)
- A strategy "is based on differentiated and clearly articulated customer value propositions. [6] (Customer perspective)
- Value is the result of well defined business processes, that need to be focused, effective and aligned. [6] (Internal perspective)
- Need of a strategic alignment, which will determine the value and role of assets. [6] (Learning & Growth perspective)
- A strategy needs to highlight the "most critical processes supporting the customer value proposition", through the "simultaneous and complementary themes" it is composed of. [6]

1.2 Blue Ocean Strategy

A Blue Ocean Strategy focuses on unexplored market space and on using the absence of competitors as an advantage to establish a dominant position in that market space. Contrary to industries with fixed structural conditions, the goal here is not differentiation or low cost, but clearly to break rules and standards and to create new ones that are more appropriate to a particular market space. [4] [5] [6]

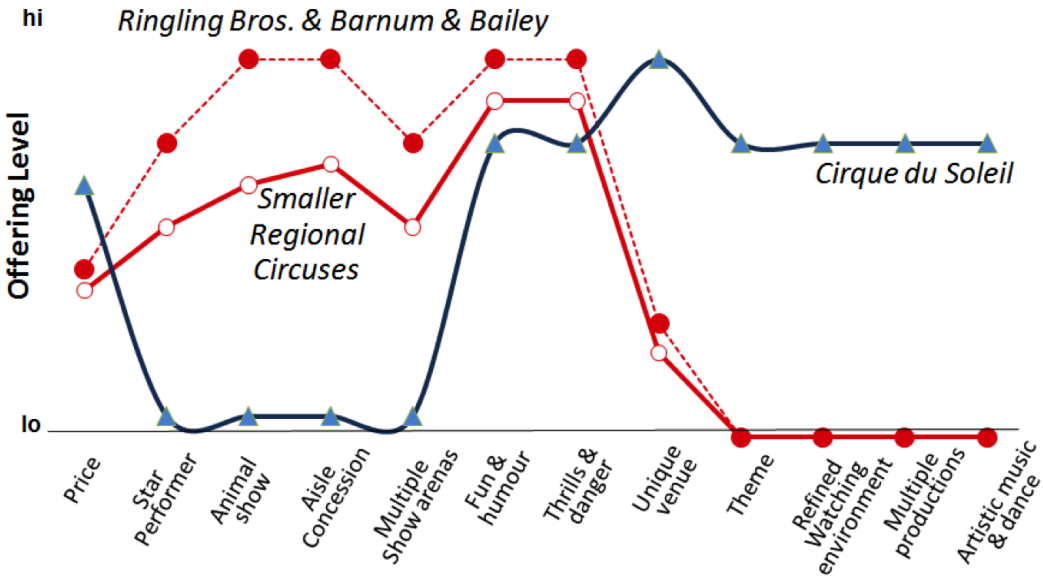

Figure 2.2: Strategy canvas capturing the BOS for the Cirque du Soleil

A strategy canvas offers a graphical representation of information relating to the current state of known markets by identifying a set of factors an industry competes on and invests in (horizontal axis in Figure 2.2[1]) and the level on which these factors are offered to the buyers (vertical axis in Figure 2.2).
One particular component of the strategy canvas is the value curve, which is an element that captures an enterprise's performance across all the factors in a strategy canvas. This strategic canvas can be derived after the analysis of the industry's market space through the techniques and tools provided by BOS. [4] [5] [6]

[1] http://bergconsulting.com.au/Berg_Consulting_Blog/what-is-blue-ocean-strategy-part-3-the-strategy-canvas

1.3 Value Configuration

The concept of Value Chain was first described by Michael Porter in 1985 and is the result of his work on competition, where he explains that there are two roads to success in a competitive environment the first one being differentiation and the second being low cost. These two ways in addition to an enterprise's desired targeted market segment, result in three strategies - cost leadership, differentiation and focus - that can be adopted. [3] [4] [6]
An enterprise's strategy as well as its implementation heavily depend on how the enterprise's activities are accomplished. Those activities are exposed in Porter's Value Chain. (Figure 2.3 [2])

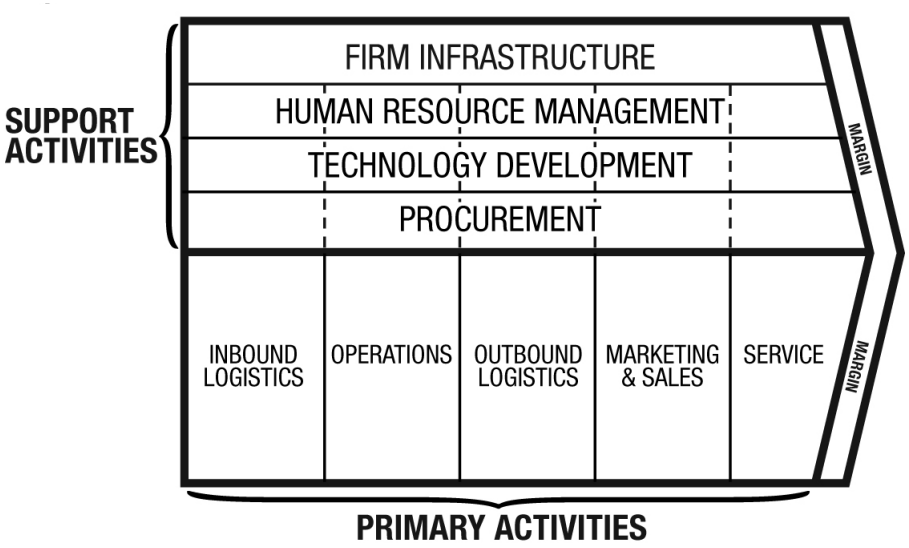

Figure 2.3: Porter's Value Chain template

The template is composed of various value activities and a margin. The value activities represent all the activities a company performs, in order to create value, and they are divided into two categories, primary and support. The margin represents the difference between the total value and the total cost of performing the value activities, meaning it is a performance indicator. [3] [4] [6]
Aside from the decomposition of activities mentioned before, activities can further be divided into three types, direct, indirect and quality attributes. Direct activities are used to create value, indirect activities allow direct activities to be performed and finally quality assurance activities are there to ensure the quality of direct and indirect activities. [3] [4] [6]
Consequently, each activity is classified, depending on the way it contributes to an enterprise's competitiveness, to, either "those that have high impact of differentiation or those that have a considerable proportion of cost". [6]

[2] http://www.thebusinessowner.com/business-guidance/business-strategy/2007/05/the-value-chain-does-yours-help-you-compete-and-win

Value activities are also able to interact with each other. Those interactions are limited to the boundaries of a Value Chain, but can exist between primary activities and between primary activities and support activities. The interactions are done through linkages, which represent relationships between the way a value activity is performed and the cost of another activity for example. They are represented through dotted lines in the template (Figure 2.3). [3] [4] [6]
The last element that Porter introduces is the notion of driver. There are ten different drivers for cost and value and "which shape the competitive nature of the firm": scale, capacity, linkages, learning, utilization, policy decisions, interrelationships, vertical integration, location timing and government regulations. [6]

The notion of Value Configuration (VC) was proposed by Stabell and Fjeldstad in 1998 and was thought of as the result of adding Value Shop and Value Network to Porter's Value Chain.
The Value Shop (Figure 2.4 [12]) focuses on the resolution of customer problems, through the use of resources and activities, to create value, whereas for the Value Network (Figure 2.5[12]), the goal is to balance cost and value, while facilitating communication between a network of enterprises and their customers and at the same time structuring the interconnected activities horizontally, seen in the Figure 2.5. [3] [4] [6]

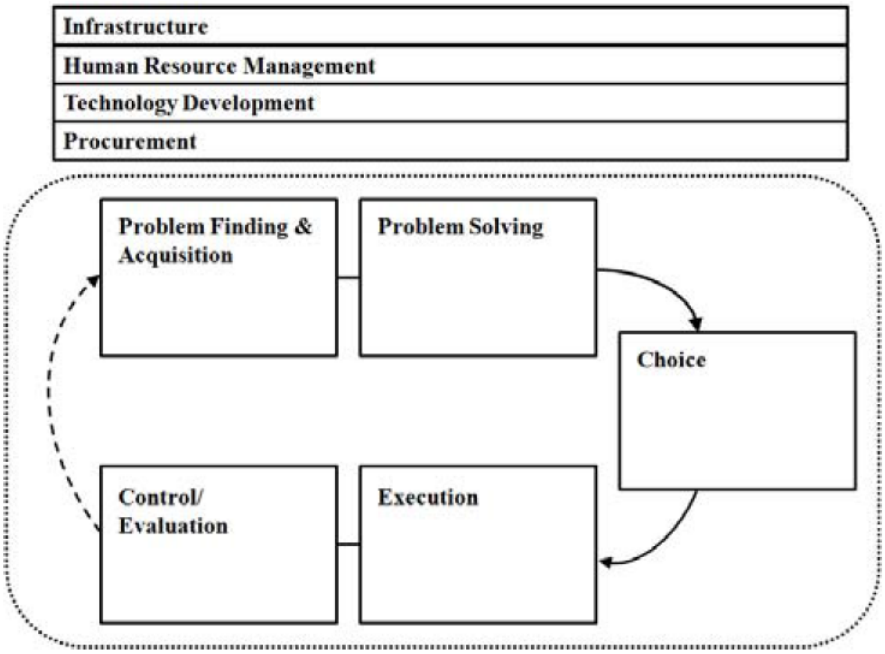

Figure 2.4: Stabell and Fjeldstad's Value Shop

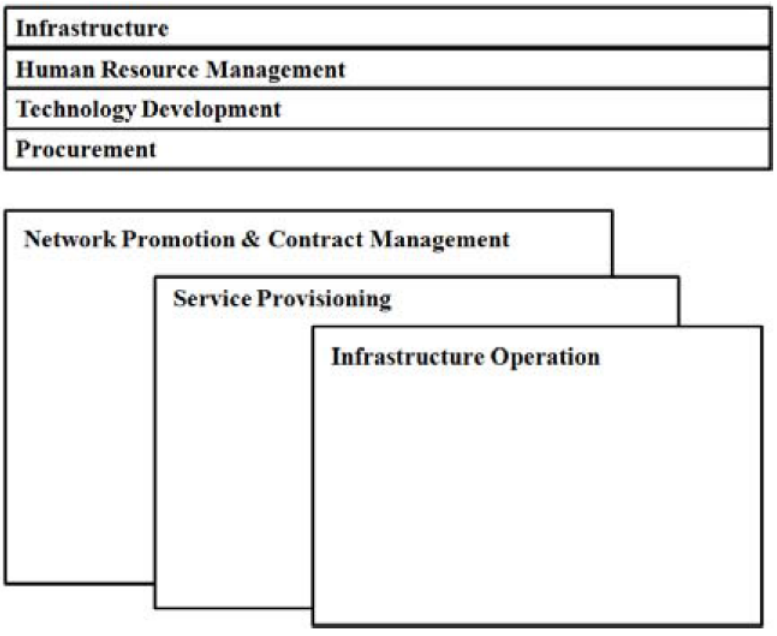

Figure 2.5: Stabell and Fjeldstad's Value Network

2 Unified Business Strategy Meta-model

The Unified Business Strategy Meta-model (UBSMM) is the result of using schema integration on various schemata. Those schemata, which are SMBSC and VC, were chosen on basis of their completeness, relevance and reliability and the fact that they are complete conceptualizations of business strategies. This is validated through instantiations of the meta-model, as well as their formalization, which gives the ability to instantiate each business strategy formulation. [5] [6] [9]
The conceptualization of the meta-model, meaning of the two retained business strategies, is represented through the use of UML conceptual models.

There exist various ways to integrate schemata. There is the binary way, for the integration of two schemata, and there is the n-ary way, for the integration of n schemata. Additionally the binary way can be decomposed into: [6]

Ladder
: Which is used when two schemata are to be integrated and another schemata is to be integrated within the result.

Balanced
: Which is used when schemata need to be integrated symmetrically, while at the same time being separated into pairs.

While the n-ary way can be decomposed into: [6]

One-shot
: Which is used when an integration needs to be done in one step.

Iterative
: Which is used when an integration is done in several steps.

The method used for UBSMM was a binary ladder integration, because of its "progressive and gradual unification" properties. [6]
The reasons why SMBSC and VC were chosen for the integration of UBSMM are, first that no other business strategy formulations had been formalized at the time and secondly that these two strategies were well recognized in the domain of strategic management. [5] [6] [9]

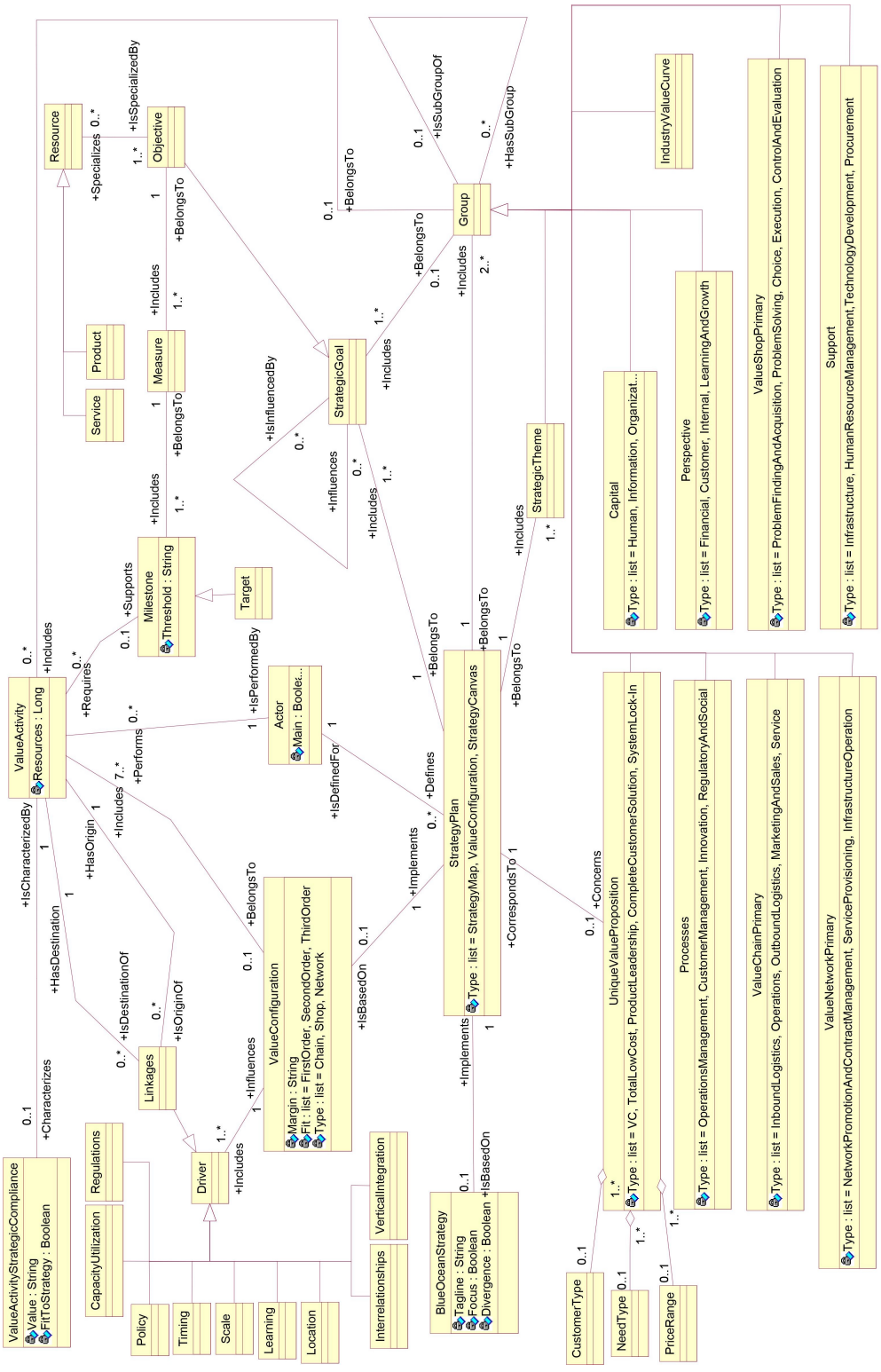

Figure 2.6: Latest version of the UBSMM [9]

3 i*

The i* version that has been used for the totality of this thesis was defined by *Samer Abdulhadi*, *Jennifer Horkoff*, *Eric Yu* and *Gemma Grau* and its complete documentation is available on the official wiki page of the RWTH Aachen University[3].

Goal modelling languages are used to capture the motivations of all the participants of a business model, as well as the objectives they wish to achieve.
i* is one of those goal modelling languages and it possesses two different models to accurately represent the elements mentioned before. The first one is the SD model, which is used to capture all the actors of a business model and the relations they may have between each other.

Actors

Actors are "active entities that carry out actions to achieve goals by exercising their know-how". *Roles*, *Agents* and *Positions* are subtypes of *Actors* and can be used in different situations to provide a more accurate representation of an entity.

Association link

There exist a number of *association* links that help describe the relationships between *Actors* by linking them to one another. The only link that we will explain is the is-part-of link, which is used to represent the fact that an *Actor* is part of another *Actor*.

Strategic dependency

A *strategic dependency* is composed of three elements. A *Depender*, which is the *Actor* that depends upon a dependency relationship, a *Dependee*, which is the *Actor* the dependency relation depends upon, and the *dependum*, which is the element the whole relation centers around. This element can either be a *Goal*, a *Task*, a *Resource* or a *Soft-Goal*, with the choice of the *dependum* changing the meaning of the dependency.

[3] http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Guide

The second one is the SR model, which describes the rationale behind the dependencies of actors. It also provides detailed information on how the different *Actors* achieve their goal, by using four elements, *Goals*, *Tasks*, *Resources* and *Soft-Goals*. There also exist various relations to connect those elements under certain conditions. All these elements are included in an *Actor boundary* of an *Actor*.

Nodes

A *Goal* is an "intentional desire of an actor", but which doesn't contain any information on how this *Goal* is to be achieved.

A *Task* is an action that an *Actor* performs, usually to achieve a *Goal*.

A *Resource* is some kind of entity that needs to be provided in order for another element to be accomplished.

The last element is the *Soft-Goal*, which is similar to a *Goal*, except that it doesn't have clear criteria defining its satisfaction.

Means-End link

A *Means-End* link is used between a *Goal* as the end and a *Task* as the means to achieve this end. The link is used to represent the fact that the *Goal* can only be achieved if the *Task* is completed beforehand.

Decomposition link

A *Decomposition* link is used to decompose a *Task* into one or many of the four types of nodes mentioned before. This decomposition can give additional information on how a *Task* is to be accomplished or can introduce new *Goals*, known as subgoals.

4 ADOxx and AdoScript

"ADOxx is the meta-modelling development and configuration platform for implementing modelling methods." The platform can be used with its administration toolkit, to implement fully functioning modelling methods, elements and libraries, which can then be used in ADOxx modelling environment to create models based on those implementations.
ADOxx is a complex environment, which allows a user to do and create many things, which is why we can't explain all the elements of ADOxx used during this thesis. We will limit ourselves to the explanations of a few crucial elements and invite the reader to refer to the website for additional information [4].

The administration toolkit of ADOxx possesses a modelling language implementation section, which is used during the creation and deletion of classes and relations as well as class attributes and attribute types. Those classes and relations are used later on, during the creation of a library and are the elements that need to be represented in models. Before they can be used in models though, they need to have shapes, which are defined in the GraphRep subsection.

Once the elements are implemented, users need to use the section about mechanisms and algorithms implementation to specify what should happen with the various elements. First there needs to be a section defining the elements that can be used during the creation of various models, since nothing indicates that all elements should be available in every type of model. This is done in the Modi window, where multiple views of a model can be added, if they are needed.
After the Modi has been completed, the behaviour of a library is implemented in the external coupling subsection of ADOxx using AdoScript.

AdoScript is a language created by the ADOxx development team of the University of Vienna and which is used to implement the behaviour of a model throughout its creation. This is done by using triggers to detect the various events happening during the creation of a model. Triggers usually include a number of command calls in their code, which always have the following structure[5]:

[4] http://www.adoxx.org/live/adoxx-documentation;jsessionid=583778B99E989FA3966CE377AAF276B6
[5] http://www.adoxx.org/live/adoscript-language-constructs

CC + "Messageport"

Every call to an AdoScript command has to start with a CC "Messageport", with messageports being internal instances of ADOxx, which decode LEO text and execute the resulting command.

API Command

The command that needs to be executed is specified in capital letters and needs to be a command compatible with the Messageport it is being sent to.

Input Values

Input values are used or not, depending on the command that needs to be executed.

Result Values

Result values are used or not, depending on the actions of the command that has been executed.

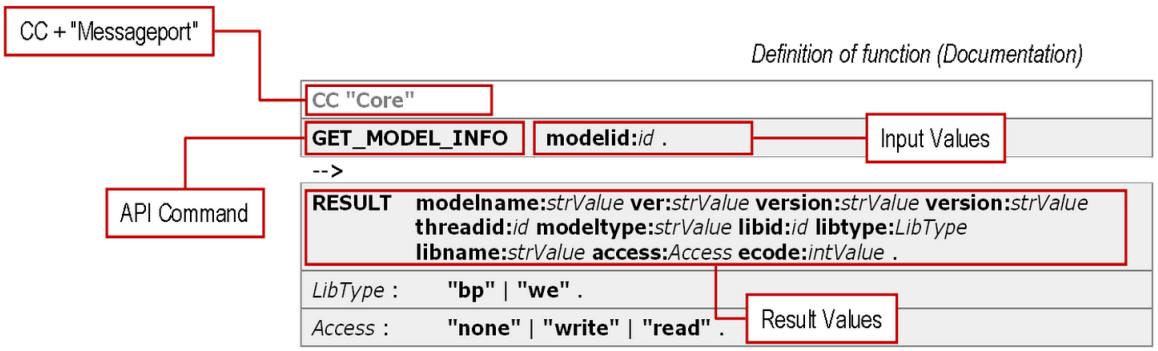

Figure 2.7: AdoScript Command Call structure

CHAPTER 3

Related work

1 Existing mappings for SMBSC

In 2011, Constantinos Giannoulis and Jelena Zdravkovic explained the mappings they had created for SMBSC in a paper at the *5th International i* Workshop*[7]. Their mappings are meticulously explained in this paper and are based on the UBSMM we mentioned before.

These mappings are regrouped in this table:

SMBSC	**i***
Strategy Map	SD & SR model
Group, Perspective	Actor
Goal	Goal, Soft-Goal
Objective	Goal
Measure	-
Target	Goal
Milestone	Goal
Initiative	Task, Plan, Resource
Theme	Critical dependencies

Table 3.1: Existing mappings from SMBSC notions to i* elements

These mappings were considered to be incomplete and some mappings violated some i* constraints. Which is why, we are going to create those mappings from scratch so as not to be influenced by the choices that were made before and trying to obtain a complete and correct mapping table.

2 Existing mappings for BOS

The mappings from BOS to i*, which have been created by Constantinos Giannoulis and Jelena Zdravkovic and which have been well documented in a paper [8], were considered to be complete and correct.

This means that the mappings in the table below are not going to be recreated, but are rather going to be reviewed in this paper and changed if there is a need to do so:

BOS	**i***
Enterprise	Actor
BlueOceanStrategy	SR model
Focus and Divergence	Dependum
Factor	Goal and Soft-Goal
Resource	Resource
NewValueCurve	SR model of the enterprise with its factors
IndustryValueCurve	SR model of the other markets from the strategic canvas with their factors

Table 3.2: Existing mappings from BOS notions to i* elements

3 i* library for ADOxx

The University of Vienna developed a library for i* in ADOxx[11] and made it, as well as its detailed documentation, available to the public.
This library was used for several reasons, the first one being that all the i* elements had been implemented through the classes and relations notions in the Modelling Language section. The classes and relations were defined in such a way, that many of the basic i* constraints were directly included during the definition of classes and relations.

The second reason was the presence of the graphical representation for all the i* elements in the GraphRep section. Every class and relation has the same graphical representation in this library as the one in the i* documentation we are using. Not needing to create all the shapes resulted in huge time savings.

The final reason was the very complete documentation[11] provided by the development team, which would help understand the library, but also the structure and functionality of ADOxx, through concrete examples.

Part II

Theoretical contribution

CHAPTER 4

Methodology

The first step for every mapping to either be created or reviewed, was to grasp and export an exhaustive list of the elements for each considered business strategy. This list of elements will then be used as the base of the whole mapping process. Before going any further in the mapping process, it is capital to have a good understanding of the three business strategies[4] [5], UBSMM[5] and i*[1]. The documents about UBSMM are used to describe the roles of the different business strategy elements and the possible dependencies those elements may have among one another. Those informations will then be used together with the knowledge of i* to create linkages between the business strategy elements and the i* elements.

The next step in this theoretical part, consists in partially validating the mappings created for each business strategy, using an i* template in Microsoft Visio, to create concrete representations of the different business strategies in i*. The limitation of the validation lies in the fact, that the Visio template doesn't check for any i* constrains, meaning that it only works as a basic representation tool and nothing more. Being able to create visual representations of the different mappings in i* is nonetheless a first step in verifying the correctness of the created mappings, since they facilitate the verification of the constrains by hand. If during this last phase any i* constraints are violated, the mappings can already be corrected a first time before moving on to the implementation of the mapping library.

[1] http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Guide

CHAPTER 5

Analysis of mappings

Since the mappings are a result of various constraints from both UBSMM and i*, we will now explain how each mapping came to be. We will start by looking at the SMBSC mappings, followed by the BOS and VC mappings. A case study will follow each explanation, where we will create a i* model in Visio based on real business strategies used by the companies ABB Industrie AG, Southwest Airways and Ikea. The mappings that were created will be applied and the resulting models will give us an idea of what the mapping models will look like. Every section will also contain a table with a summary of the mappings from a business strategy to i*.

1 Analysis of SMBSC

As we mentioned before, the mappings of this section were created from scratch, since the analysis of the mappings done before my arrival showed various problems. This doesn't mean that the newly created mappings can't be similar to the already existing mappings.

1.1 Explanations

The first notion from SMBSC that will need a mapping i*, is the *Strategy Map*, which is a particular type of *StrategyPlan* in the UBSMM. The issue here, is to fully understand what conditions a *Strategy Map* has to fulfil to be correct, before we jump to any conclusions.

A *Strategy Map* contains at least one *Strategic Goal* or *Objective* in each of the four *Perspectives* of SMBSC, meaning that we need to consider the fact that the element representing a *Perspective* in i* needs a way to contain the i* element representing a *Strategic Goal*. This condition greatly limits us in the mappings we can consider, since there aren't many i* elements allowing such a behaviour. The best choices for those mappings are, in our opinion, to map *strategic map* to the **SD** and **SR models** and to map a *Perspective* to a **role**. The **SD** and **SR models** capture the whole i* model, which includes the perspective mappings, which, since they are mapped to **roles** have actor boundaries attached. The mappings of *Strategic Goal* or *Objective*, we will analyse shortly, will then be placed inside those boundaries.

What we need to do now, is map a *Strategic Goal* to one of the many elements allowed inside this boundary and which is somewhat near the meaning of a *Strategic goal*. After careful consideration, a **soft-goal** is the best mapping for this notion, since it also represents a type of goal and we wanted to differentiate it from the mappings of *Objectives*, *Milestones* and *Targets*, which we are going to analyse further down the road.

When we talk about an *Actor* in SMBSC, we talk about the company that is using SMBSC, which means that there really is only one actor at all times. Since we are already using **roles** for earlier mappings, it would seem wise to try and use **roles** as often as possible to facilitate the comprehension of elements used in the model. Mapping the *Actor* to a **role**, allows the connection, using is-part-of connector, between this role and the roles of the four perspectives. This representation varies from a previous representation, in that it doesn't violate any i* rules. The main issue with the mappings that were given to me at my arrival, was the fact that the representation was very crowded and not i* compatible. The trouble came from the use of i* actors, representing the *Perspectives*, within the *Actor*'s boundary, which violated numerous i* constraints. To correct this problem, the externalisation of the **roles** from the boundary, meaning from the SR model to the SD model and linking them using the **is-part-of** connectors to the main role, seemed like the best solution. Additionally

to the already existing **roles**, the notion of *Group*, which is a super-type of *Perspectives*, is also mapped to **roles**, to keep the mappings coherent with the UBSMM.

In SMBSC, a *Strategic Theme* is considered to be a set of interrelated goals, which may have influences to one another, be it within the same perspective or even through different perspectives. This last condition is the one that dictates the mapping, since the only element of i* that allows inter-perspective influences is a **dependency link** and to be in accordance with the i* constrains we are going to link every **goal** and **soft-goal** to the corresponding **goals** and **soft-goals** from other *Perspectives*. These dependencies can either be goal dependencies or soft-goal dependencies, depending on the nature of the elements that need to be connected.

This leaves us with influences between goals within the same perspective, to be more specific they are within the boundary of the role representing this perspective. The issue with those influences is, that we can't represent them through **dependency links**, since these links aren't allowed between elements of a same actor. The solution is to externalize the goals depending on one another to new **roles** and link them to the **role** of the *Perspective* through **is-part-of** links. With the goals being in different **roles**, **dependency links** can now be used to show the influences and we haven't changed the belonging of the goals to the original perspective thanks to the **is-part-of links**.

The **goals** we just mentioned, can be a representation of SMBSC *Objectives*, *Milestones* or *Targets*. The motivation to use **goals** for *Milestones* and *Targets* is simply that they essentially are SMBSC goals that need to be achieved before a *Strategic Goal* can be accomplished. Concerning *Objectives*, the UBSMM tells us that they are subtypes of *Strategic Goals*, meaning that they are a more detailed version of the *Strategic Goals*. As seen before, *Strategic Goals* are mapped to **soft-goals**, which are similar to **goals** except that the criteria for their satisfaction are not clear-cut. As a direct consequence, if the *Strategic Goal* is detailed enough, we can consider it to be an *Objective*. The same procedure is used for the mapping since a **soft-goal**, that possesses strict criteria of satisfaction, isn't considered to be a **soft-goal** but rather a **goal**, which finally leads us to the mapping of *Objectives* to **goals**.

The link between *Objectives* and *Milestones*, or *Targets*, which are a subtype of Milestones, is made through the notion of *Measure*. A *Measure*is an indication on what has to be achieved for an *Objective*,*Milestone* or *Target* to be met. The issue is, that there is nothing in i* that a *Measure* can be correctly mapped to, which is why we have to resort to a somewhat unconventional solution.

The way we are representing *Measures* is through a single **task**, since it can be linked to **goals** in i*. The result of this mapping may be syntactically correct but unfortunately, we can't say the same about the semantics, with the meaning of a *Measure* and a **task** being so different. The lack of a better solution forces us to use this suboptimal solution to integrate *Measures* in i*.
The reason why we have to choose such an i* element, is because of the association links between *Milestone* and *Measure* and between *Measure* and *Objective* in the UBSMM, which translates in *Measure* being mapped to an i* notion, that can be linked to at least two goals. The **task** is linked to the **goal**, representing an *Objective* through a **means-end link** and is decomposed through **decomposition links** to a number of **goals**, representing either *Milestones* or *Targets*. The number of **decomposition links** originating from the **task** is equal to the number of different *Measures* for the considered *Objective*.

Furthermore, each **goal** mapping a *Milestone* is the target of a **means-end link** originating from a **task**, which corresponds to the mapping of SMBSC's *Value Activity*. This last mapping is quite intuitive, with a *Value Activity* capturing all the actions required to achieve an objective. The same can be said of a **task** linked to a **goal** through a **means-end link**, where the **task** needs to be completed in order for the **goal** to be accomplished. By adding a **resource** to the **task** representing a *Value Activity*, we complete the mapping of this element by adding the time and money needed to accomplish the *Value Activity*. We will apply those mappings to an example in the next section of this chapter, where further explanations will make it easier to understand.

1.2 SMBSC for ABB

To start the case study of ABB Industrie AG[1][1], we create roles for the four different perspectives (Financial, Customer, Internal, Learning & Growth) connected to a central role (ABB) through is-part-of links. This will be the core of the whole i* model. For the customer, internal and learning & growth perspectives, we created some more roles that are linked through the same type of links to the perspective roles. Those roles are either needed for the creation of dependency links between objectives that belong to the same perspective and need to be externalized or they are optional roles, that we added for better readability of the model.

Next we are going to work our way through the different perspectives, starting with the financial perspective. This perspective has 3 main objectives and each objective has milestones and measures. We are going to talk about objectives instead of strategic goals, since they are described at a level, that allows us to consider them as objectives. As an example we will be mapping the objective "our net margin is constantly >15%", since the same rules apply for the other objectives. First we create a goal, which has the same name as the objective, then we create a task. This task is linked to the goal by a means-end link and is the first part of the mapping for this objective's measure.

This task is decomposed into a number of goals, which is equal to the number of measures for an objective. In this case, the objective only has one measure, so the task will only be decomposed to one goal. This goal corresponds to the target of the objective, meaning the last step before the accomplishment of the objective. The target is achieved through the completion of a value activity, which is mapped to a task, that is linked to this goal through a means-end link and decomposed to a resource and another goal if there are more milestones for this objective. This step is to be repeated as many times as there are milestones left for an objective.

Perspectives	Strategic Goals	Measures	Milestones		Target
			End of 1st year	End of 2nd year	End of 3rd year
Customer perspective	We offer an innovative service concept	Number of realised process optimisations	1	3	5
		Share of sold systems with additional service contracts	15%	30%	40%

Figure 5.1: Extract of the Customer Perspective requirements of the "Printing Facilities" Business Unit of ABB Industrie AG

[1] http://www.imsciences.net/wp-content/uploads/2012/04/Applying-the-Balanced-Scorecard-Concept-A-Case-Study-of-ABB.pdf

For the customer perspective all the steps are the same as the ones for the financial perspective, except that the objective has two measures for one objective, which means that we need to create one more decomposition link from the task to a goal. To sum up, the result of the mapping of the two measures is now a task with two decomposition links. Each link connects to a target and the next steps are the same as before. Mapping of value activities, followed by milestones, repeated as long as there are milestones left to map. Here is the result of the mapping for the customer perspective:

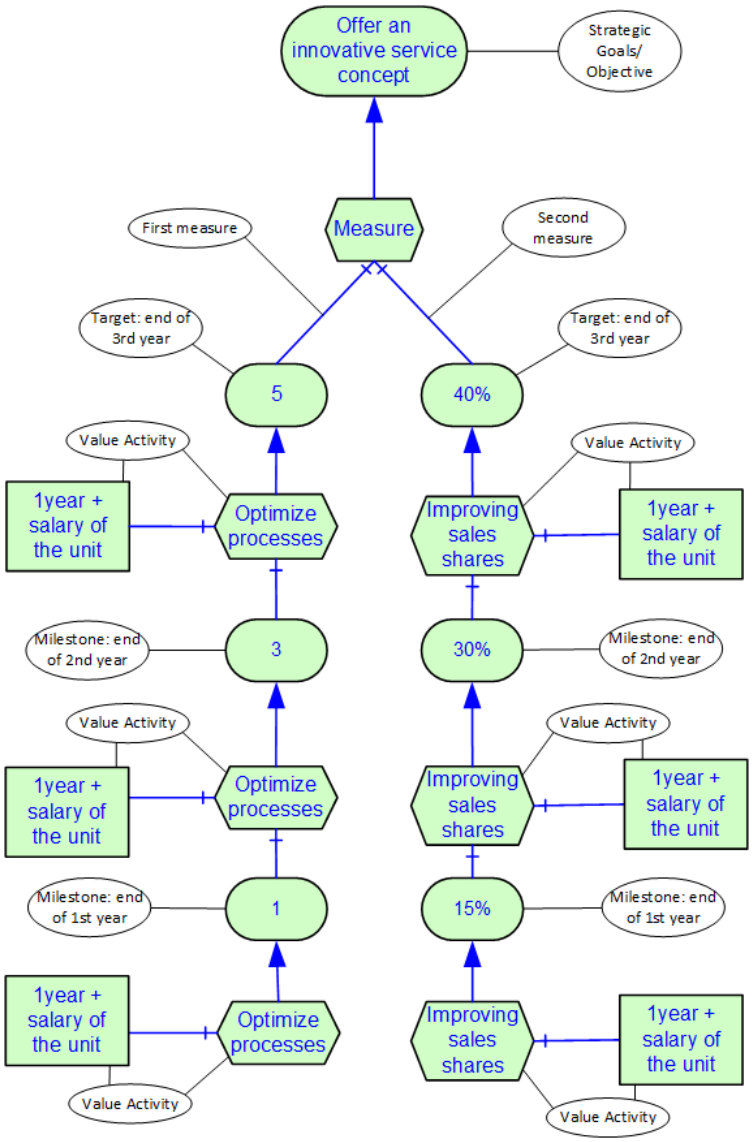

Figure 5.2: Result for the mappings of Strategic Goal, Measures, Milestones and Target for the Customer Perspective of ABB Industrie AG in i*

As said before, the exact same rule is applied for all the objectives of all the perspectives, which only leaves us with the mapping of the strategic theme of this case. This is done by

using dependency links and dependums between roles, more precisely between the goals of those roles. To create those links, we are looking at the strategic goal network (figure 5 of document[1]). This figure shows which dependencies exist between the different goals from different perspectives. Applying those dependencies on this model is quite easy, since the dependencies only exist between goals from a lower level perspective to goals from the next highest perspective. The hierarchy of the perspectives is the same as the one in figure 5 of the document, with financial perspective being the highest.

For every goal of the role "financial", we draw dependency links to the corresponding goal in the role "complete customer solution", with the name of each goal in role "financial" chosen as the name for each goal dependum for the dependency links. The same is done between role "complete customer solution" and roles "process1" and "process2". Finally the last dependency links that need to be created are the ones from roles "process1" and "process2" to roles "capital1" and "capital2". The determination of the names of the dependums always follows the same rule for each dependency link in this model.

1.2.1 Visio model for ABB using i* template

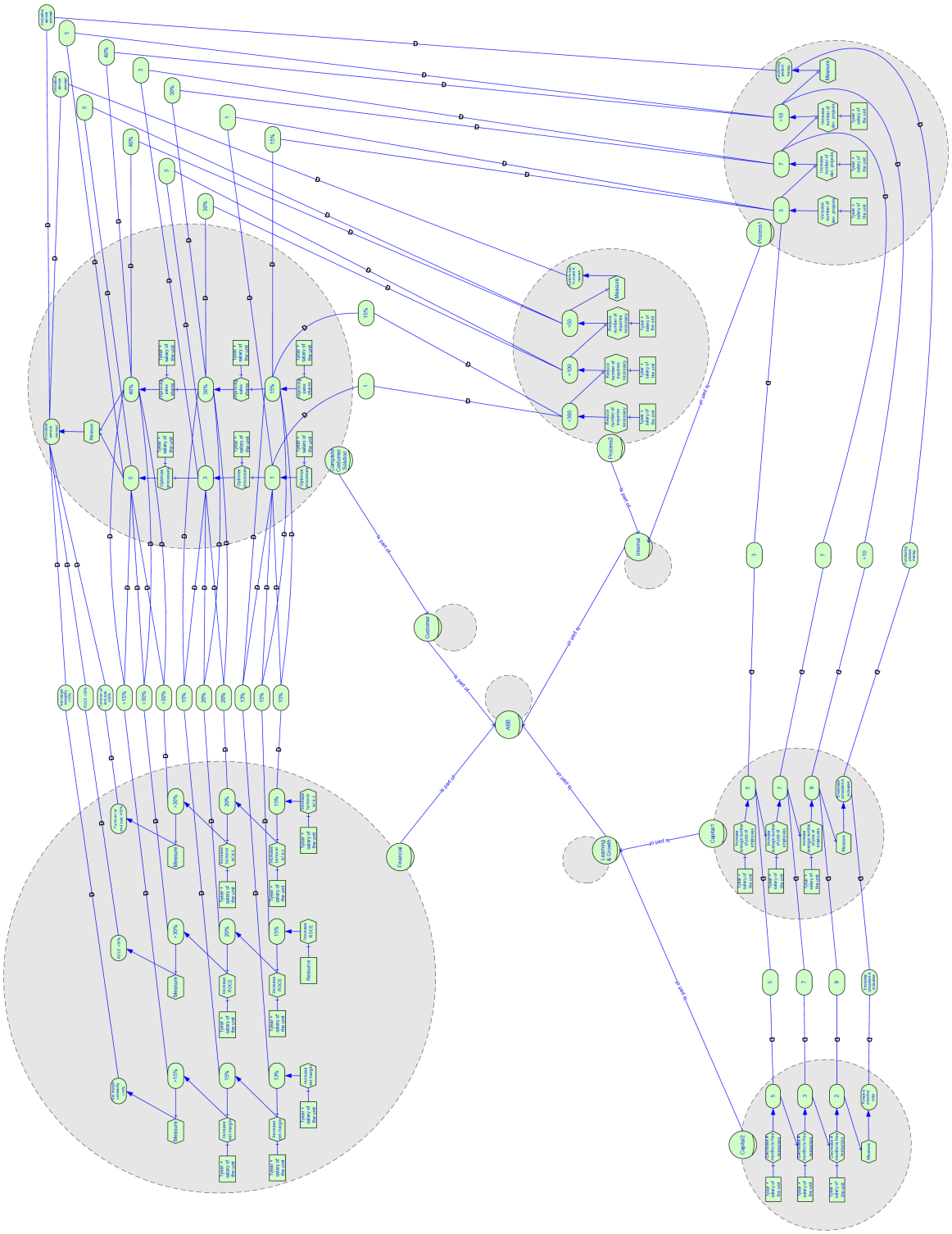

Figure 5.3: ABB's SMBSC in Visio (detail of each perspective available in the appendix)

1.3 Mapping table for SMBSC

This table regroups all the latest mappings from SMBSC elements to i* elements, we have discussed in the last chapters. Those mappings are not necessarily optimal, but they correspond to the latest findings of our research and can be applied to create accurate i* models representing Strategy Maps and Balanced Scorecards.

SMBSC	**i***
Actor, Group, Perspective	Role
Strategy Map	SD & SR model
Strategic Theme	Critical Goal dependency with a Dependum
Objective, Milestone, Target	Goal
Strategic Goal	Soft-Goal
Value Activity	Task with a Resource attached
Measure	Combination of a task and a number of branches, regrouping the Milestones and Target, departing from that task. The number of branches is equal to the number of different measures for a particular Objective

Table 5.1: Created final mappings from SMBSC notions to i* elements

2 Analysis of BOS

2.1 Explanations

Before going any further in our analysis of the mappings, it is essential to explain why it seems like BOS isn't really integrated in the UBSMM. The reason is, that the BOS concepts are similar to existing elements of UBSMM. Reading section 7.1 on page 25 of the document "Schema Integration Process for UBSMM" [5] will shed some light on the conceptual correspondences between BOS and UBSMM and help understand the mappings that we will now explain.

The first element that will interest us, is the notion of *Enterprise*. The meaning of this element is pretty straightforward, since it represents the enterprises, that are concerned by a Blue Ocean Strategy. This notion is best represented by a **role**, since we will need to include different elements of BOS, like factors for example, inside its actor boundary to show their belonging to the different *Enterprises*. The next element we are going to look at, is the strategy itself *BlueOceanStrategy*, which is represented through an **SR model**, since we need to represent the fact that the strategy may depend on the actions of other enterprises, shown through *Focus* and *Divergence* in BOS. There is but one way to represent these two elements and that is through two goal **dependums** linked to each role of the other enterprises the main enterprise compares to, thanks to **dependency links**. We are going to represent the fact that the strategy is focused, through the first goal dependum, while the second goal dependum is divergent to allow comparisons between the *NewValueCurve* and the *IndustryValueCurve*.
We will come back to the dependums further down the road, once we have introduced the elements that will be linked through **dependency links**.

"The *NewValueCurve* captures the value curve created by applying the four-action framework for the enterprise whose strategy is modeled/represented" [8]. This definition mentions the value curve, which is the graphical representation of the enterprise's performance across the different factors of competition it possesses, meaning that the *NewValueCurve* needs to be mapped to an element that can contain the mappings of the *Factors*. Those *Factors* represent states that somehow need to be achieved, but for which 1) no information relating on how to realize those states is available or 2) no way of verifying, whether a state has been realized or not, exists. This means, that we can map *Factors* to two different i* elements, depending on the case it relates to. We either map it to a **goal**, if there is no information on the way to achieve a state, or to a **soft-goal**, if there is no way to verify the realisation or not of a particular state. Now that we have the mappings for the factors, we can decide on a mapping for the *NewValueCurve*. After careful consideration, mapping the notion to an **SR model** seems like the obvious choice, since we need to represent the fact that the factors belong to the *NewValueCurve* and this can be done if we

consider the mapping to be an **SR model** containing the **goals** and **soft-goals** for the factors.

The mapping of *IndustryValueCurve* is the same as the one for *NewValueCurve*, since the enterprises, the main enterprise is in competition with, capture the same type of elements. Mapping these elements, meaning the *Factors* to either **goals** or **soft-goals** and placing them inside the **SR models** of the different competitors of the main enterprise, leaves us with the final tasks of linking the different goals of the enterprises and the mapping of *resource*. Since there exists a **resource** element in i* with a similar meaning to the BOS notion, we are going to map one resource to the other.

The final step consists of linking the main **goal** of the main *Enterprise* to the main **goal** of each other *Enterprise*, through **dependency links** and the aforementioned **dependums**. The number of links originating from the main *Enterprise*'s goal is equal to double the number of competitors and each link possesses a **dependum**, so that we have a focused goal **dependum** and a divergent goal **dependum** in the **dependency links** to each of the competitors' main goals.

2.2 BOS for SouthWest Airways

This model is made quite easily by using the "Strategy Canvas of Southwest Airlines" [8] (figure 5.4) and applying the mappings. First we create roles for Southwest, Average Airlines and Car Transport, with their respective actor boundaries. We will first start with the role Southwest, since the mappings can be used on the other roles afterwards. First step is the creation of the elements representing the "NewValueCurve", a task linked to a goal through a means-end link. The goal of Southwest is to provide Airline Services with “The Speed of a Plane at the Price of a Car – Whenever You Need it” and the task is of course to provide this service. The task is then decomposed to a resource "Airline Services" and to a number of goals and soft-goals representing the factors.

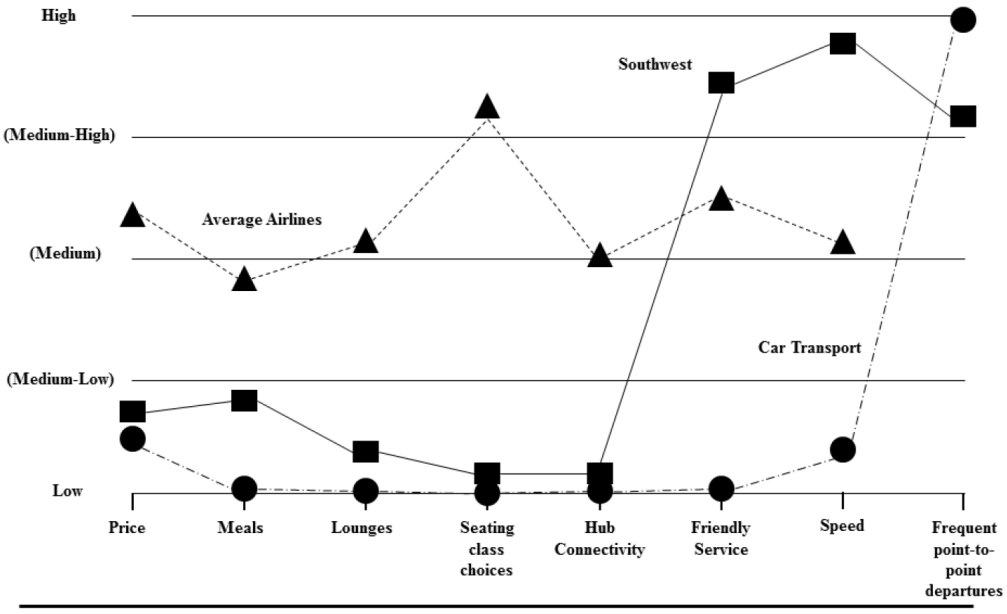

Figure 5.4: Strategy canvas of Southwest Airlines

The next step is giving those goals and soft-goals names, which is easily done by checking, where the dot of a factor is located on the strategy canvas. If the information is at hand, we can add tasks using means-end links to the different goals, to give a more complete model.

After applying the same method to the two *IndustryValueCurves* for the two remaining roles, the only thing left to do is link the different roles, since Southwest relies on the doings of "Car Transport" and "Average Airlines". We create dependency links between the main goal of role Southwest and the main goals of roles "Average Airlines" and "Car Transport". We will need to create four dependency links, with different dependums between these roles. We will need a first dependum representing the fact that it needs to be focused and a second one representing the fact that it needs to be divergent, for each competitor. This means, that we will have a first dependency link with *Airline Services with "The Speed of a Plane at the Price of a Car – Whenever You Need it" Provided be Focused"* as a goal dependum going towards the goal "Car Services be provided" and a second link with *Airline Services with "The Speed of a Plane at the Price of a Car – Whenever You Need it" Provided be Divergent* as a dependum going to the same goal. We finish by doing the same procedure towards the goal "Average Airline Services be provided", which completes the model.

2.2.1 Visio model for SouthWest Airways using i* template

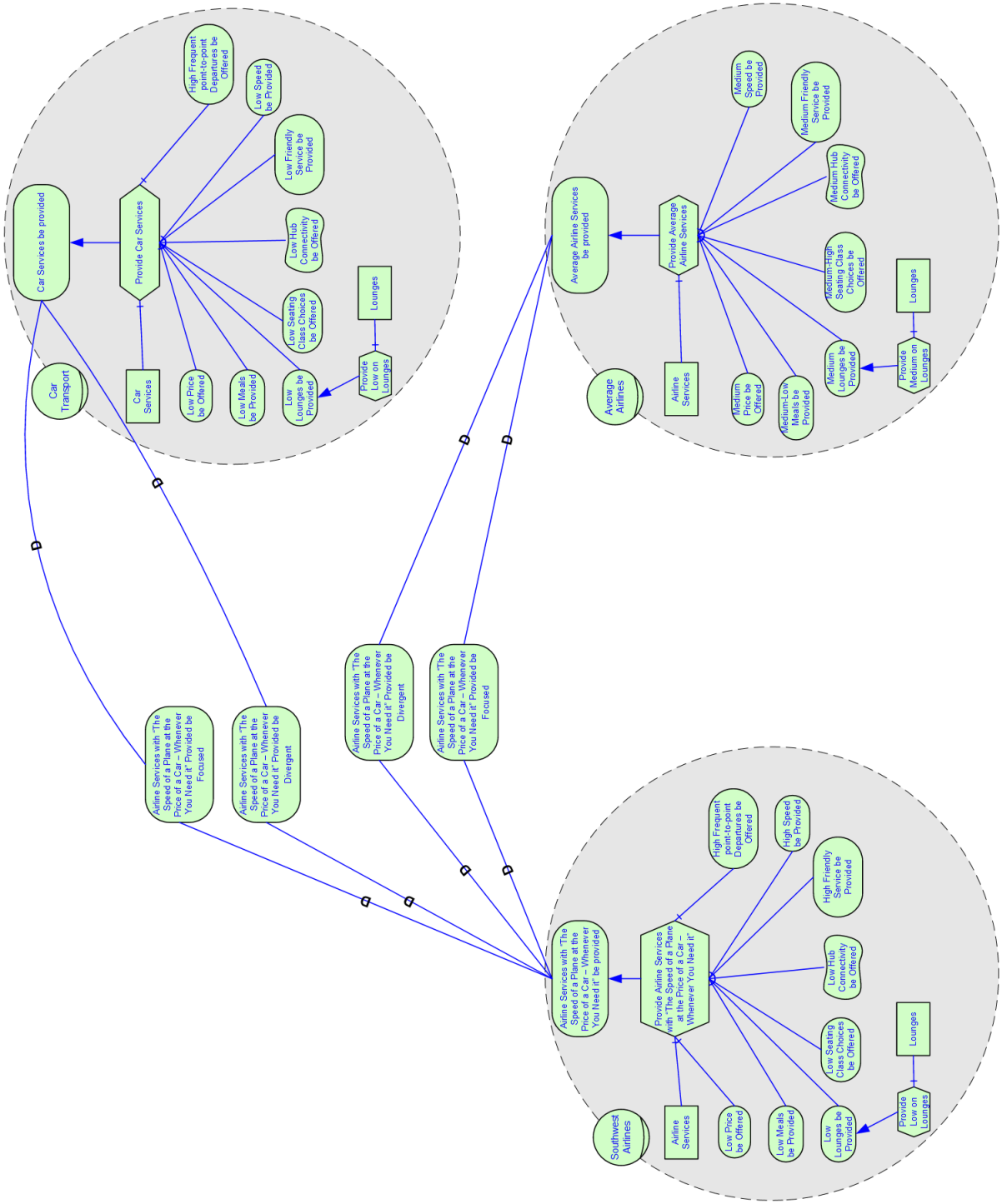

Figure 5.5: SouthWest's BOS in Visio

2.3 Mapping table for BOS

This table regroups all the latest mappings from BOS elements to i* elements, we have discussed in the last chapters. These mappings were created by J. ZDRAVKOVIC and C. GIANNOULIS and reviewed in this thesis. They are not necessarily optimal, but they correspond to the latest findings of our research and can be applied to create accurate i* models representing a Blue Ocean Strategy.

BOS	**i***
Enterprise	Role
BlueOceanStrategy	SR model
Focus, Divergence	Goal Dependency with a Dependum
Factor	Goal, Soft-Goal
Resource	Resource
NewValueCurve	SR model for the Enterprise with its factors
IndustryValueCurve	SR model of other markets from the strategy canvas with their factors

Table 5.2: Created final mappings from BOS notions to i* elements

3 Analysis of VC

3.1 Explanations

The first two mappings we are going to look at, are somewhat linked and this is because *Value Configuration* is a specific part of the *Strategy* itself [5]. It captures the configuration of elements that contribute to the value of the strategy, based on one of three different cases, which are either Value Chain, Value Shop or Value Network. Depending on the type of configuration used in the strategy, we have different *Primary* activities, that need to have at least one *Value Activity* for each such activity. There are also the *Support* activities, which remain the same whatever the configuration used and also need at least one *Value Activity* for each such activity. For example in the case of Value Chain, there needs to be at least one *Value Activity* in each of its primary activities, we talked about in chapter 2, but also at least one *Value Activity* in each support activity, which are the common activities among the different configurations.

Now, if we first map the *Strategy* to the whole model, meaning to both the **SD** and **SR model**, we will need to find a mapping for *Value Configuration*, that somehow is more restricted than the *Strategy* and also includes the mappings of *Value Activities* inside the *Primaries* and *Supports*. To be able to do so, we will need to use the **SR model** again, simply because there is no other way of representing the fact that those *Value Activities*, which will be represented in different actor boundaries, as we will see shortly after, are a result of the *Value Configuration* that was chosen.

The next element that we are going to map is the *Actor*. The definition of an *Actor* is very similar to the one of i*'s **role**, meaning that we are going to continue in the same fashion we did until now and use this mapping. A **role** represents the company using the strategy, like we had for SMBSC and BOS and it will once again be the destination of all the **is-part-of links** from the surrounding **roles** we will define shortly. *Group* is another familiar notion from the previous mappings and as with *Actor*, we will once again use the same mapping used for the other strategies, meaning a **role**.

As we know, VC is a combination of Porter's Value Chain and Fjeldstad and Stabells' Value Shop and Value Network, meaning that there needs to be a way to represent all three of those strategies, preferably through the same i* elements. Before we start explaining why we chose to represent the activities of *ValueChain*, *ValueShop* and *ValueNetwork* through **roles**, let us remind ourselves of what their differences are.

ValueChain

ValueChain captures all the activities of a Value Chain, which is a strategy that focuses in the transformation of various inputs into products. The primary activities Inbound Logistics, Operations, Service, Marketing & Sales and Outbound Logistics, as well as the support activities Infrastructure, Human Resource Management, Technology Development and Procurement.

ValueShop

ValueShop captures all the activities of a Value Shop, which is a strategy that focuses on the resolution of customer problems. The primary activities Problem Solving, Choice, Execution, Problem Finding & Acquisition and Control & Evaluation, as well as the support activities Infrastructure, Human Resource Management, Technology Development and Procurement.

ValueNetwork

ValueNetwork captures all the activities of a Value Network, which is a strategy that focuses on the networking and interaction between customers. The primary activities Infrastructure Operation, Service Provisioning and Network Promotion & Contract Management, as well as the support activities Infrastructure, Human Resource Management, Technology Development and Procurement.

The idea for the mapping of these strategies is to start by creating a **role** for *Primary* and *Support*. These are used to be even more precise in the representation of VC in i*. They are going to be used as intermediates between the main role and the various activities. Primary activities are going to be linked to the *Primary* role and this role is going to be connected to the main role. Support activities are going to be linked to the *Support* role and this role is then connected to the main role. All the connections we just mentioned are going to be done through **is-part-of links**. This finalises the mapping of the activity structure. The last element we are going to map to a **role** is *UniqueValueProposition*.
This element captures the way the actor delivers unique value thanks to three specific **goals** inside *UniqueValueProposition*'s boundary. These goals are *CustomerType*, *NeedType* and *PriceRange* with the first one capturing customer information, the second one capturing need related information and the last one capturing price relevant information. Before going any further with in our analysis, we need to introduce the **goal** inside the actor's boundary representing a *Strategic Goal*. This mapping is due to the fact, that the *Strategic Goal* is a precise objective, which makes it very similar to an i* **goal**. Additionally the *Strategic Goal* also depends on the three goals mentioned before, which we are going to "regroup" in a soft-goal through **means-end links**. This soft-goal then becomes the dependee in the **dependency link** with the representation of the *Strategic Goal* of the *Actor* playing the depender of this dependency. There may seem to be some strange mappings in this last paragraph and lets be honest some of these last mappings aren't entirely correct in i*, which is why we are going to explain some of the compromises we had to make.

If we consider the definition of *UniqueValueProposition* it may seem strange to represent it through a **role**. The reason for this choice was impacted by the need to represent the fact that *CustomerType*, *NeedType* and *PriceRange* need to belong to the element we were going to choose to map *UniqueValueProposition* to and there is no other choice in i*, but to represent it the way we did. Once this choice was taken, we had to make compromises on the representation of the dependency between the **goal** representing the *Strategic Goal* and the **soft-goal** regrouping the goals. Usually it is not recommended to use a dependency link between two different elements, but since there was no way to regroup the goals in another goal, because of the fact this goal would need to have a very specific objective, which we can't simply create out of nowhere, we decided to go with this idea. To be clear, the way we used the dependency link isn't forbidden in i*, it is just a practice that is not recommended and since we only use it for this particular case and for notions that remain somewhat similar, we fell comforted in our decision to represent those notions through these mappings.

Before looking at other dependencies, we will first finish the mappings of the elements filling the actor boundaries of the activity roles. The way each activity creates value is identified by its *Value Activity*. We talked about these elements in the section about the *Value Configuration*in chapter 2, where we explained the need for at least one *Value Activity* for each of the *Primary* or *Support* activities. Using **tasks** to show what needs to be done to achieve some value and adding *ValueActivityStrategic Compliance* as a **resource** to those tasks, is an easy and correct way to represent these notions.
This last mapping is the result of being able to link a **resource** to a **task** and to represent who, what or how the task is achieved and in being able to do so, we found a mapping with a very close meaning to *ValueActivityStrategic Compliance*, which gives an explanation on the way a value activity brings value.
Finally we are going to finish the mappings of this section, with the ones for *Driver* and *Linkage*, the latter being a specific type of driver. A driver gives information on the cost and value of an activity inside a VC. This means, that translating a driver into a **dependency link** linking the activity and the actor by way of a dependum, is the only way to go. The dependum is a **resource** to give information on what resources the roles depend on, except in the special case of the driver being a *Linkage*, where we need to use a task dependum. *Linkages* are different from the other *Drivers*, since they represent dependencies between activities and no longer between an activity and the actor.

3.2 VC for Ikea

The case we are going to apply the VC mappings on, is the Value Chain for Ikea [2]. This means that Ikea is the Actor of the model and it will be mapped to a role in i*. This role will be considered as the main role of the i* model.

The next elements that will be added are the ValueChain activities, since we are using a Value Chain model for Ikea. Every activity, primary or support, is mapped to a role. These roles could be linked directly to the main goal, but this wouldn't be accurate enough, since we wouldn't know which activity is primary and which activity is support, from looking at the model. To do so, we are going to create a role for Primary, which will be the destination all the is-part-of links originating from the roles representing primary activities of Value Chain.
Another role is created for Support, which will be the destination all the is-part-of links originating from the roles representing support activities of Value Chain.
The roles for Primary and Support are then also linked through is-part-of links to the main role. This results in the model representing, the fact that certain activity roles belong to the primary role and others to the support role, but also that they all belong to the main role and thus the Actor of this Value Chain.
The names for all the roles representing activities were mentioned in the previous section.

Next we are going to map the notion of UniqueValueProposition to a role and link it directly to the main role through an is-part-of link, showing that this role and everything related to it, is part of the Actor.

Since there is no specific strategic goal for this case, we can assume, that such a goal for an enterprise like Ikea can be resumed to "Superior Longterm Return on Investement be Achieved". This is represented by a goal in the boundary of the main role, since it is the goal the enterprise wishes to achieve.
This goal depends on the realisation of the soft-goal in the UniqueValueProposition role, we needed to introduce, to regroup the notions of CustomerType, NeedType and PriceRange. These elements are mapped to goals and linked through means-end links to the soft-goal. This means that all three goals need to be achieved for the soft-goal to be accomplished. Once the soft-goal is in the desired state, it influences the goal of the main role.

We are now going to take a look at what the value activities for all the activities of this Value Chain. Each value activity is represented through a task and is placed into the boundary of the activity role it belongs to.
We will only show the detail of those mappings on one of Ikea's activities, since the same process can then be applied to all the other ones. The activity we are going to look at

[2] http://research-methodology.net/ikea-value-chain-analysis/

is "Operations", which is a primary activity, with three value activities. The first one is "operate its stores directly", the second one "operate stores through franchises" and the third one "no own product manufacturing". These three value activities are mapped to three tasks and placed inside the activity role for the activity "Operations". As mentioned before, the same procedure is applicable for the remaining activities and value activities.

The last elements of a VC model, that we would need to add to the model to make it complete, are the drivers and linkages. The issue is that there is no information on either of them in the case used, meaning that we won't be able to complete the model. What we are going to do, is explain how linkages would be represented, if information about their existence was available. Since linkages represent the fact that an activity or the role representing this activity, depends on the actions of another activity or the role representing this activity, we would need to use a dependency link between the depender role and dependee role of the model and add an adequate dependum, which would then accurately represent the linkage between the elements.

3.2.1 Visio model for Ikea using i* template

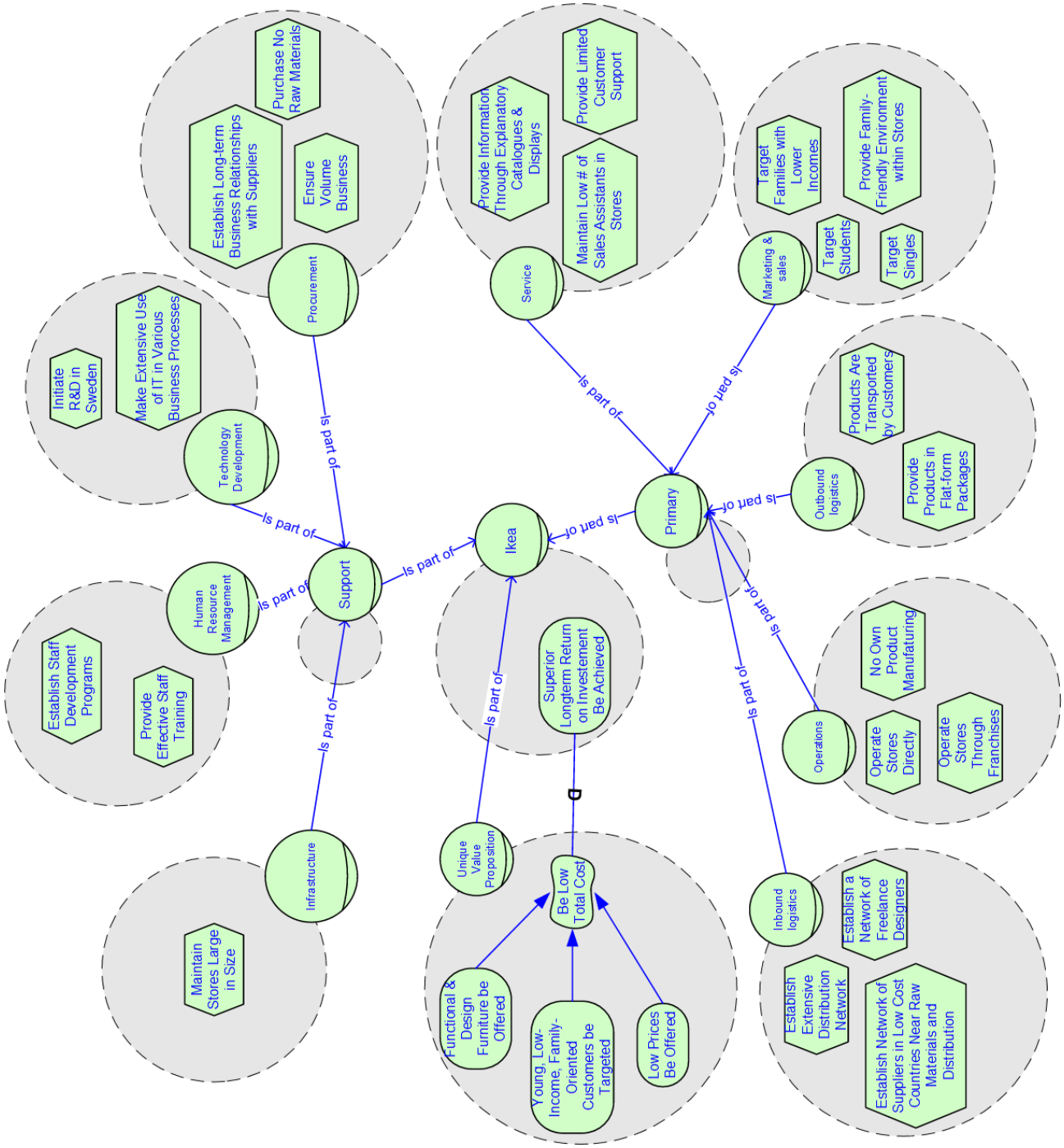

Figure 5.6: Ikea's VC in Visio

3.3 Mapping table for VC

This table regroups all the latest mappings from VC elements to i* elements, we have discussed in the last chapters. Those mappings are not necessarily optimal, but they correspond to the latest findings of our research and can be applied to create accurate i* models representing a Value Configuration.

VC	**i***
Strategy	SD and SR model
Value Configuration	SR model
Actor, Group, ValueChain activities, ValueShop activities, ValueNetwork activities, Primary, Support, Unique-ValueProposition	Role
Strategic Goal	Goal
Value Activity	Task
ValueActivityStrategicCompliance	Resource (attached to a Value Activity through task decomposition
Driver	Dependency Link (with a Dependum)
Linkage	Task Dependency (Task as Dependum)
CustomerType, NeedType, PriceRange	Goals within the UniqueValueProposition Role

Table 5.3: Created final mappings from VC notions to i* elements

CHAPTER 6

Intermediate evaluation for the theoretical part

Understanding the different business strategies and all the i* constraints has led us to a set of mappings. We had to make some compromises during the creation of the mappings for some more complicated notions, but the resulting mappings appear to be correct. We use the word "appear", since at this point there is only one way of verifying the mappings and that is by hand. We looked at each business strategy notion and tried to find an i* element that could represent it. We looked at dependencies and links between the notions and chose the mappings accordingly.
We then created the models for various cases, which allowed us to have a graphical view of the mappings and have an easier way of verifying the mappings by hand. We didn't notice any major issues with our mappings, which gave us a first, if not very formal, confirmation of the validity of our findings.
The next part of this thesis will look at providing a better library for creating the mappings and with the added i* constraints, give us a way to better verify the correctness of our mappings.

Part III

Practical contribution

CHAPTER 7

Requirements

After having created and reviewed the mappings, the next step was to create a library for those mapping in the ADOxx environment. The reason for using the ADOxx environment is firstly that there already existed a library for UBSMM which was created by the University of Stockholm in ADOxx and secondly that there already existed a basic i* library [11] in ADOxx, created by the University of Vienna. This last library being available to the public, was a huge time saver, since it allowed us to directly start with the implementation of the mappings we had created before. What we wanted to create with this library, was a basic mapping modelling tool, meaning the only i* elements that should be available, are the ones used for the different mappings and we will discard the remaining elements.

Once the desired library is created, it will be used to recreate the models we created in Visio before and check if the mappings have been correctly implemented in the library, but also check if the created models are i* sound. This last statement is the most important one, since until now, there was no other way to check the i* validity of the models, except by doing it by hand, which isn't a very efficient way of doing this. So having a library regrouping the i* constraints and our mapping constraints is the ultimate goal of this section.

CHAPTER 8

Design choices

The library we wish to create has to be user-friendly and complete. The ideas for fulfilling the first condition are, firstly to have some mapping constraints generated automatically, to facilitate the use of the library and save time while creating a model. Secondly, to be able to switch easily between an empty model, if the user wants to create a model from scratch, and one tab for each i* view, SD or SR model, for each of the three business strategies, resulting in a total of seven "tabs". These different "tabs" for the business strategies, should have default models, which we wanted to create to save the users some time. The idea to create those default models, came from the realization that all the i* models created for a particular business strategy, have a similar backbone structure and that it would save users a lot of time if those structures were created by default. So the SD model for SMBSC has a particular default structure and its SR model also has a default model, and the same goes for the SD and SR models of BOS and VC.

CHAPTER 9

Implementation of mappings in ADOxx

The structure of ADOxx is not the easiest environment to understand, which is why we are going to explain what every used section does in addition to the explanation about the AdoScript code. We are first going have a look at the *modi* and *external coupling* sections, as well as the kind of models we want the users to be able to create. We will then continue with a more detailed view of the AdoScript code for each business strategy.

The elements we will implement are the ones mentioned in the previous chapter. We will start by creating the different "tabs", followed by the creation of the default models for each of the "tabs". Once we can easily change between the different "tabs" and default models, we are going to add a few automated actions for the creation of different elements.

For this section we are going to try and stay at a pretty high level of abstraction, but sometimes give glimpses of the detailed implementation of various elements, without overwhelming the reader.
The sections below are all contributions made during the internship and elements that already existed in the i* library from the University of Vienna will be described as such.

1 Modi and a first look at the attributes

```
1  INCL "Role"
2  INCL "Goal"
3  INCL "Task"
4  INCL "Resource"
5  INCL "Softgoal"
6  INCL "Boundary"
7  INCL "Dependency Link"
8  INCL "Means-end Link"
9  INCL "Decomposition Link"
10 INCL "Association Link"
11
12 VARIANT "Strategic Dependency Model for SMBSC"
13 EXCL "Means-end Link"
14 EXCL "Decomposition Link"
15 VARIANT "Strategic Rationale Model for SMBSC"
16
17 VARIANT "Strategic Dependency Model for BO"
18 EXCL "Means-end Link"
19 EXCL "Decomposition Link"
20 VARIANT "Strategic Rationale Model for BO"
21
22 VARIANT "Strategic Dependency Model for VC"
23 EXCL "Means-end Link"
24 EXCL "Decomposition Link"
25 VARIANT "Strategic Rationale Model for VC"
```

The modi section of the library attributes is where we can define which elements should be accessible or not in a variant, depending on the variant chosen by a user. A variant is a specific view a user can choose to be in and which is either an empty model by default or an SD/SR model for SMBSC, BOS or VC. They represent the "tabs" we talked about before and we will explain why we decided to use this notion instead of the standard *Mode* notion, but lets start by explaining the small code section above.

Since the default variant, meaning an empty model, a project shows when it is created, will need to let users create everything from scratch, we are going to include all the i* elements for the creation of the mappings and ignore the other elements included in the library. We represent the restrictions of the elements to be used in the models, through the use of the inclusion (INCL) functions from lines 1 to 10. Since we are using the i* library created by the University of Vienna, we don't need to create the types of elements or their graphical representations, which leaves us with the simple definition of the elements we are going to need through the inclusion functions.
We then create the six variants, one for each model, by giving them detailed names of the models they represent. The variants representing the SR models for SMBSC, BOS and VC should also allow the use of all the elements needed for the creation of the mappings

and nothing more. By default all the variants use the elements that are included, which for the variants representing the SR models is perfect, but causes problems for the variants representing the SD models.
For those last variants, we need to exclude the elements representing links, that can be used inside the actor boundaries of the different actors. We only need to retain the elements that can be used in or for inter-actor linkages. This results in the exclusion (EXCL) of the means-end and the decomposition links in each variant representing an SD model, because they can only be used inside actor boundaries and aren't allowed in SD models.

Now that we have explained the modi of the library, lets come back to the reasons that drove us to using variant instead of *Mode*, which is the standard way to change between views. We actually started by creating the models in the standard views, but it didn't work as intended. The default models for each variants weren't only visible in the desired views, which resulted in all the default models being present in all the views and overlapping on one another. After searching in the ADOxx documentation and contacting the development team in Vienna, we found the reason why the models acted this way. The reason was that *Mode* views are limited to the metamodel/class level, which unfortunately doesn't allow us to work on instances and thus doesn't allow us to create the default models in the way we wanted. *Variants* on the other hand operate on instance/object level, meaning that it should now be possible to build default models in different views.

At this stage if we were to make a model with various elements in variants, we could still see elements from other variants appear in the current variant. This problem is due to the fact, that a variant can't by itself differentiate which elements belong to which variant and it results in the overlapping of the elements when switching variants. At this point, switching from *Mode* to variants doesn't seem to have changed anything, but we will now introduce the solution to the problem. To solve this issue, we need to store information about the allocation of an instance to a particular variant, as well as the exact coordinates needed to place each instance correctly in the variant it belongs to. The attribute we need to add to all the classes and relations is of type LONGSTRING and is named "___Variants___". Here is a small example of what the attributes of a class or relation look like:

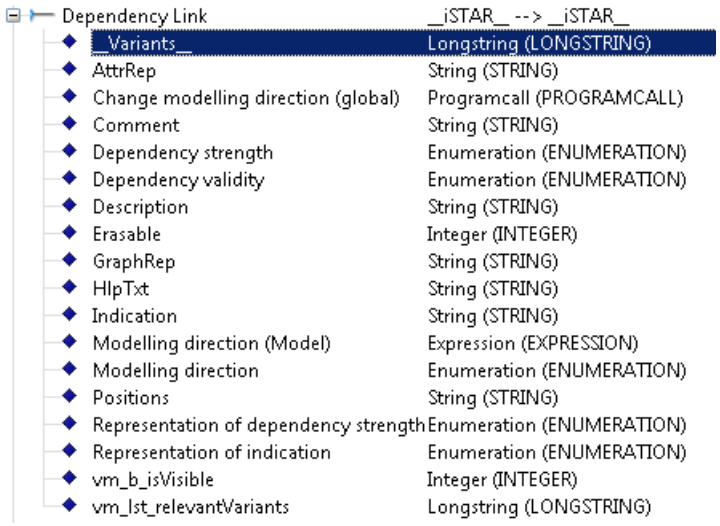

Figure 9.1: Attributes for the relation "dependency link"

As we can see, there are a number of different attributes for each class and relation and since a large part of those are generated during the creation of either a class or a relation and aren't relevant in the explanations we want to provide, we won't explain them in detail. In the next section we will look at the attributes that we added ourselves and which will have important roles in the creation of the default models and the behaviour of those models.

2 Preparations for the creation of the default models

We talked about the fact that we wanted to create one default model in each of the variants representing the SD model and SR model and do this for each business strategy. The result being one default model for the SD model of SMBSC and one default model for the SR model of SMBSC and so on, for the other two business strategies.
To be able to do so, we started by adding a "___Variants___" attribute to all the classes. This one attribute isn't enough to be able to position the different elements correctly in the variants, but it allows us to store information about the variant an element belongs to. The information about the variant first needs to be retrieved and then stored in this attribute.

We now need to introduce two small pieces of AdoScript, to go and retrieve the information about the variant an instance belongs to and then add it to the "___Variants___" attribute we just mentioned. We will only look at the AdoScript for the classes, since we only need to make small adjustments to make these changes compatible for the relations. The attributes of the classes or relations that contain the information about the position of an instance will need to be extended, because they aren't precise enough. By default they don't give any information about the variant an instance needs to be positioned in.

To remedy this, we add the small statement "iuv:0" at the end of all the attributes of the instances that are created. This statement retrieves the information about the variant from the system and stores it in the "___Variants___" attribute we created before. This fixes the problem, where instances showed up in variants they weren't supposed to be in, by providing additional information on the place an instance has to be and limiting an instance to one and only one variant.

```
%The code below is only a fragment of the trigger! We left the parts not
    related to the problem at hand aside!
ON_EVENT "AfterCreateModelingNode" {
  SETG class_id: (classid)
  SETG obj_id: (objid)
  SETG nCreatedModelID:(modelid)
  CC "AdoScript" FREAD file:     ("db:\\createInstance.asc")
  IF (text = ""  OR  ecode != 0) {
   CC "AdoScript" ERRORBOX ("Error loading the CreateInstance file")
  }
  ELSE {
   EXECUTE (text)
  }


%The code below is issued from the "createInstance.asc" file
CC "Core" GET_ATTR_VAL objid:(obj_id) attrname:("Position")
SETL sPostionInformation:(val)
CC "Core" SET_ATTR_VAL objid:(obj_id) attrname:("Position") val:(
    sPostionInformation + " iuv:0")
```

Since we haven't spoken about the code until now, we will quickly explain what each element does so as to have an idea of the basic functionality of AdoScript. Once again, we won't explain or show the relations part of the code, since it is only a slight variation of the code we will explain now.
The changes we want to make to the position of an instance need to be made after its creation, which explains the use of the trigger *ON_EVENT "AfterCreateModelingNode"*. The use of this trigger is logical, since we first need to create an instance before even thinking about storing information about its variant or get its position through the position attribute.

The next three elements of the code describe the variables used to retrieve and modify the positions of the created instances. The variable "nCreatedModelID" gives us the identification number (ID) of the whole model in which the instance was created, whereas "class_id" and "obj_id" give us the ID of the class that has been created and for the latter the ID of the created instance. The only information needed at this point is the value of "obj_id", because it is used in a command call (CC) to get the value of the attribute "position" of the instance "obj_id". The other variables are used either for debugging purposes or for the retrieval of other information later on. Before going any further, we will quickly explain the structure of a CC on these examples, since they constitute a crucial part of the functionality of the library.

To remind ourselves of the structure of CC, please refer to the explanations in chapter 2.

In our case, the first command we want to execute is the "GET_ATTR_VAL", which as one might have guessed, gives us the value of an attribute for a particular instance. We give this command two inputs, firstly the "obj_id" of the instance and secondly the exact name of the attribute we need the value of, in this case "Position". The command then returns an *ecode* of type INT with an error code if there was an error or 0 if there wasn't and *val*, which in this case is a LONGSTRING containing the value of the attribute "Position" and which we allocate to the local variable sPositionInformation.
The next command aims to set a new value for a particular attribute by adding the aforementioned statement "iuv:0" at the end of the position of the instance. We translate this by using the command "SET_ATTR_VAL" and giving the "obj_id" of the instance we want to fix to a variant, as well as "val", containing the concatenation of the position of the instance and "iuv:0", as input values.

Now that we have a basic understanding of how AdoScript works and we managed to find a way to associate the instances to variants, we will introduce a way to create default models for each business strategy and look at a way to make them permanent later on. The code for these parts being quite long, we will add it in the appendix and limit ourselves to explaining the reasoning behind the chosen default models.

3 Default models for SMBSC

Before creating the default models for the SD and SR variants of SMBSC, we have to isolate the different elements of an SMBSC model that don't depend on the case they are used in and which we will then be mapped to i* and represent the default model.

The first element that is always present in any SMBSC model, is an Actor. We mapped this element to a role in i* and since it is the central part of the whole model, we will create this role in the default model for both the SD and SR variant of this business strategy.
To display this element and every other element, when a user navigates to either the SD or SR variant, we need to use a trigger called "AfterCreateModelWindow" to make something happen once a a new model window has been created. In this case we want to create a role and its boundary for both the SD and the SR variants at the initialisation of the model, but the same procedure is used for the other default elements.
Once the trigger is activated, we start by setting the current variant to the SD variant for SMBSC and we create a role and its boundary in this variant. During the creation of those elements, we need to hardcode their positions and add the "iuv:0" statement we introduced a few sections back, to set the "___Variants___" attribute of the elements to this variant. Since the left and upper borders of the model window are fixed and the model can't be expanded that way, we set the positions of the elements more towards the lower right, so that bigger models can easily be created, without having to change the location of the whole model. Once this element is created, we change the current variant to the SR variant of SMBSC and we proceed the same way.

Now that we have our main role, we look at other elements that are represented in every SMBSC and we notice that every such model has the four perspectives. Once again the mappings we created before lead us to representing the perspectives through roles with their respective boundaries. The way to represent those roles in both SD and SR variants for SMBSC is the same as we discussed before.
These new roles need to be linked through is-part-of links to the main role, as we explained in during the mappings chapter. We create instances of those relations and specify which roles are connected by each link. Other than this minor difference, the remainder of the procedure to create and limit them to the correct variants is the same.

For the SD default model, this seems to be all that can be added, without making assumptions on the cases that will be created in this variant. For the SR variant on the other hand, we could think about adding a mapping for a strategic goal in each perspective, but since it is such an important element of SMBSC, we won't make any assumptions concerning this element and leave its positioning and creation to the user.

With these elements included in the default model of the SMBSC for both variants, we have a small but useful model, that will help the user understand where the remaining elements of SMBSC need to be placed.

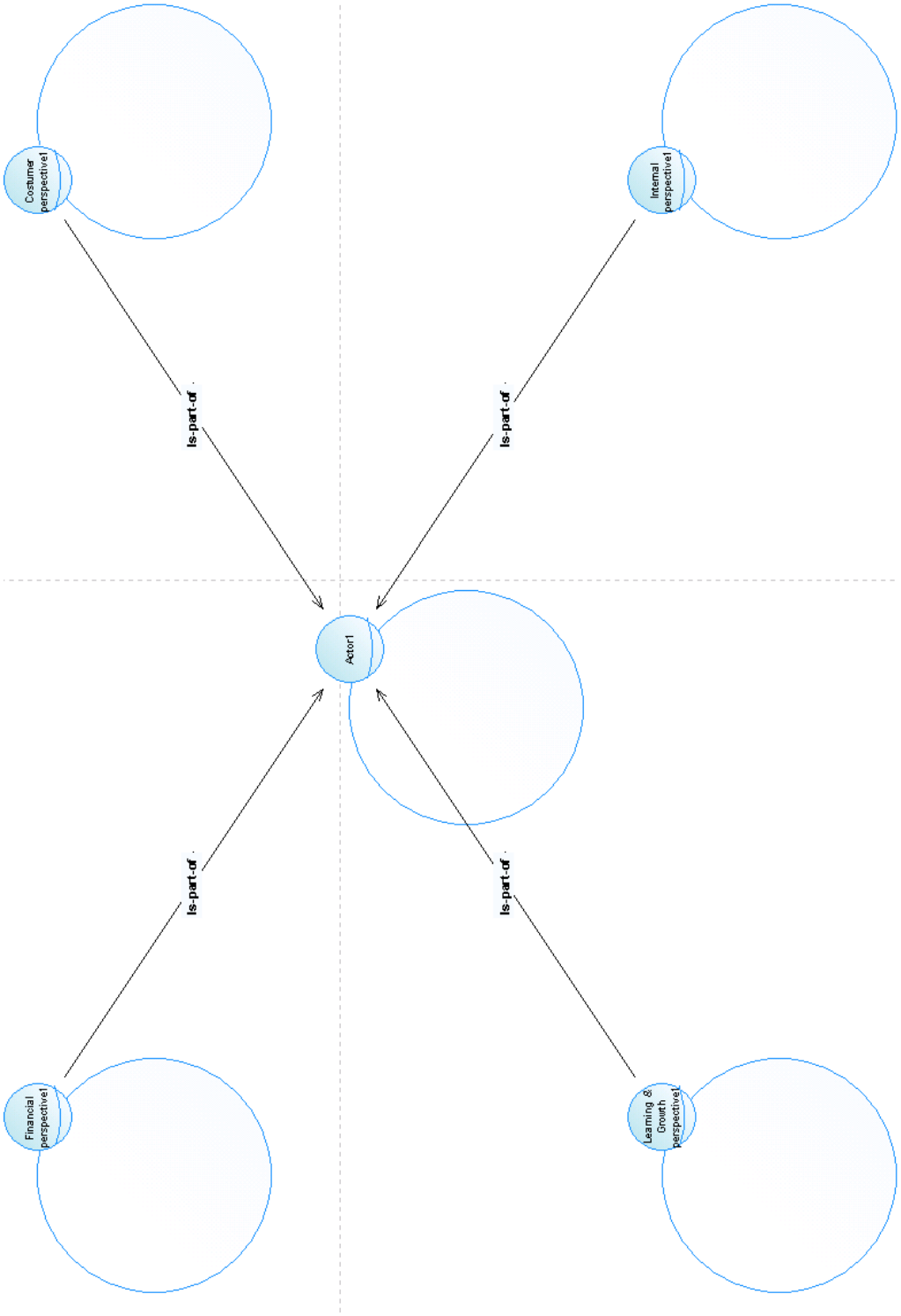

Figure 9.2: Default SD and SR models for SMBSC in ADOxx

4 Default models for BOS

Similarly to what we did for the default models in SMBSC, we are going to check which elements should or shouldn't be in the default model for BOS models.
We are going to start with the examination of the enterprise element from BOS, which, since it represents the enterprise the strategy is used on, will surely be present in every BOS created. The steps to adding this element to the default model are the same as the ones we described in the section before, with the only difference being the variants, the role and its boundary will be added to. We do the same for two other enterprises, since BOS can only be applied if several enterprises are considered.

After having introduced three roles in the default model, we have to introduce the Focus and Divergence notions to be consistent with the constraints of BOS. Here is the first time we have a different default model in the SD variant from the one in the SR model.
On one hand, in the SD variant, we will have two dependency links between the main role and each of the other roles, with one goal dependum forcing the need to be focused and the other one forcing the need to be divergent. In the SR variant on the other hand, we will have to use the same dependency links from the main goal of the main enterprise to each of the main goals of the other enterprises. The dependums also remain the same. Once again these elements will be added to one variant and then to the other one.

The default model for the SD variant is finished with the addition of those last elements, but as we just saw, we need to add the main goals of the enterprises and a task through a means-end link to the main goal in the SR variant. These assumptions seems very reasonable, since every enterprise will have a main goal and a task to fulfil this task.
With the addition of those elements in the SR variant, we finally have a complete default model for the SD and the SR variants, which will help the user in the creation of the i* models representing BOS.

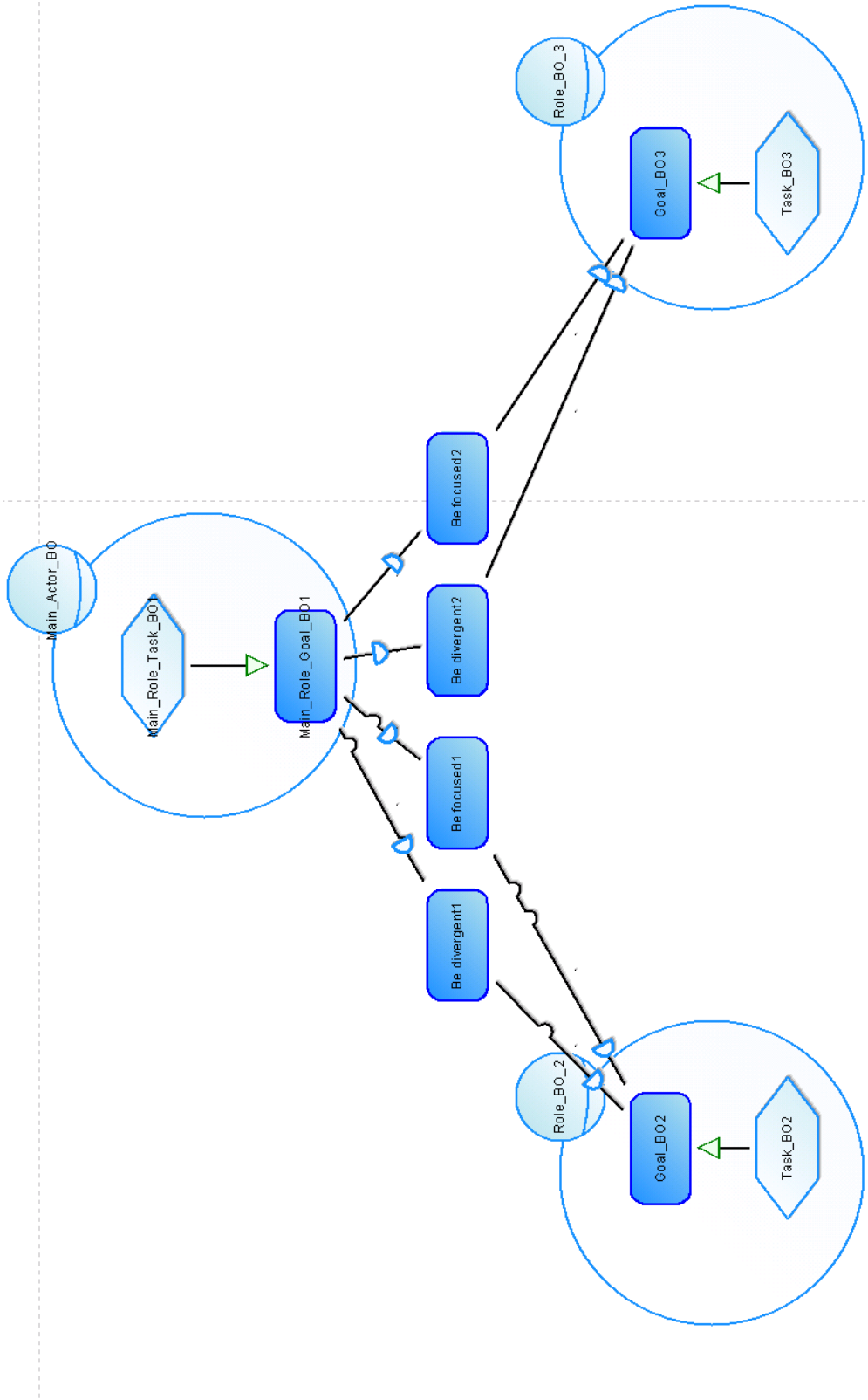

Figure 9.3: Default SR model for BOS in ADOxx

5 Default models for VC

Once again the approach for creating the default model for this business strategy is to browse through the model elements of VC and retain some of them and ordering them in such a way, that they represent the backbone of any VC model that needs to be represented in our i* library.
For this business strategy, we are going to start with the creation of the SD variant default model, since the SR variant's default model will be the same default model, with a few added elements inside the boundaries.

The first element we are going to look at, is once again the actor, the enterprise this business strategy applies to. This element is mapped to a role and its boundary and will be the center of the backbone model for any further elements we decide to add.
If we proceed further, we realise that by definition every model in either Value Chain, Value Shop or Value Network has the same support activities. This translates into the creation of a number of roles equal to the number of those support activities, which will be linked to an additional role to show that they are support activities. The link used to show this notion is the is-part-of link.
To sum up, we are going to have roles named Infrastructure, Human Resource Management, Technology Development and Procurement and link them using is-part-of links to the Support activity role and finally link this role through the same link type to the main role.
Adding the primary activities isn't an option since they change depending on the strategy used.

Next we take a look at the UniqueValueProposition, which resulted in a mapping that wasn't very intuitive and which needs to be present in every model. These two reasons make it important to add the role representing this notion to the default model. Here again we are first looking at adding it to the SD variant and talk about the SR variant addition soon after.
The role representing the UniqueValueProposition is connected to the main role through an is-part-of link.
This last addition completes the default model for the SD variant, since the remaining elements that could be added, can only be added inside the actor boundaries, meaning in the SR model.

And this is what we are going to look at now. We talked about the CustomerType, NeedType and PriceRange and the way we planned to represent them inside i* models with the help of a soft-goal. We also talked about the dependency between the strategic goal and the elements we just talked about. These elements are the only remaining ones that we retained as fixed elements for every VC model, which is why we are going to add them to the SR variant's default model.
We are going to start by creating the goal representing the strategic goal inside the main

role's boundary, followed by the soft-goal inside the UniqueValueProposition role, that will regroup the other goals and a dependency link from the newly created goal to the soft-goal. Meaning that we only have to create the last three goals inside the UniqueValueProposition role and connect them to the soft-goal through means-end links.

With these additions done, we now have default models for each variant, but the use of those models is limited as long as we haven't made them undeletable, since they could compromise the ease of use of the tool as well as the understanding of the model by the users. Which is why the next chapter will quickly explain how we managed to implement this constraint.

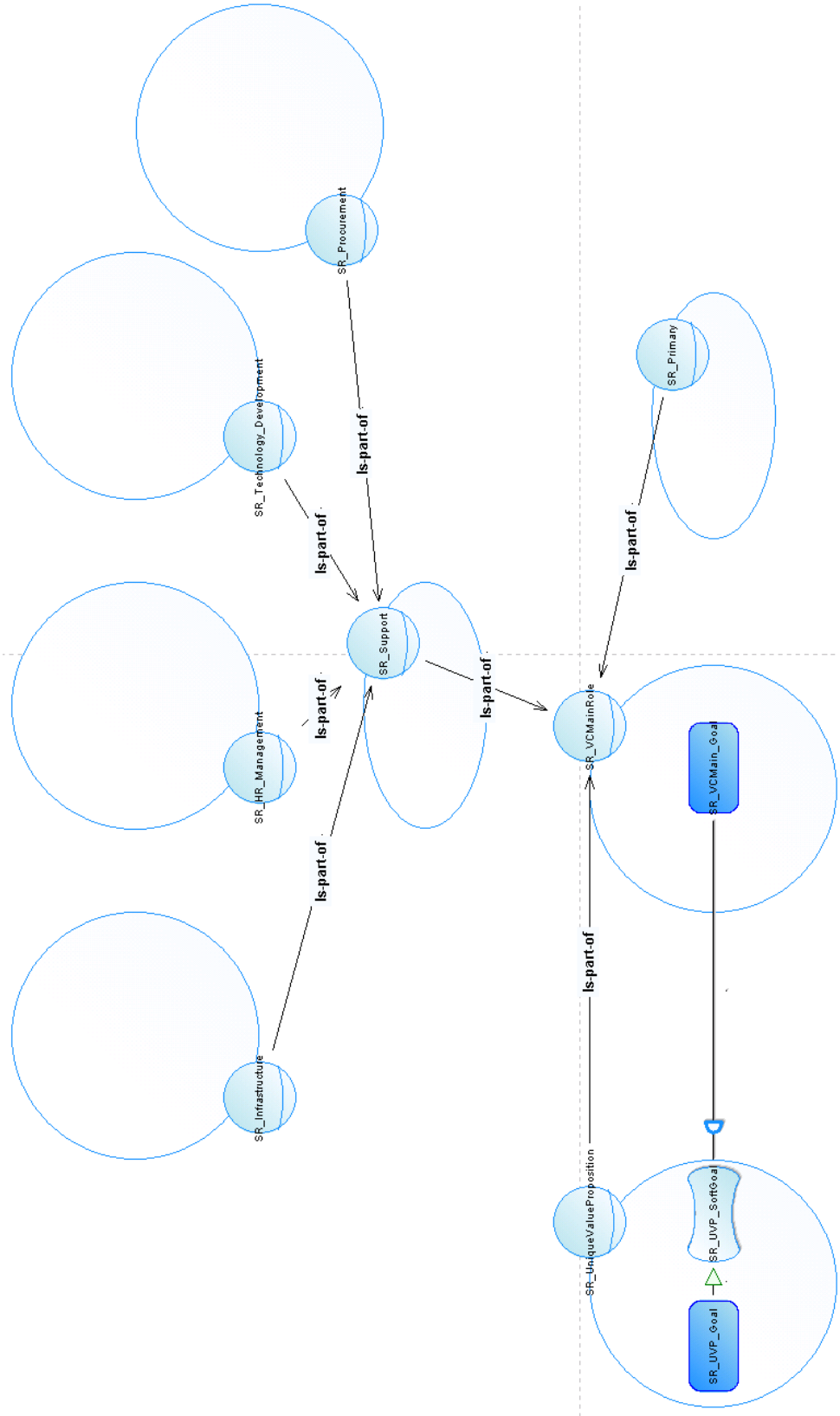

Figure 9.4: Default SR model for VC in ADOxx

6 Making the default models permanent

As we said earlier, now that we have the default models for each variant, making these models undeletable is the only remaining task we wish to implement to finally have backbone models to work on.
After carefully searching in the documentation of ADOxx to find an easy way to make instances immutable and finding nothing that could help us, we contacted the department responsible for the creation and the maintenance of ADOxx to confirm this information. The team confirmed that there is no way to tag instances and make them permanent, but instead recommended the use of different triggers.

The triggers that were recommended are "BeforeDeleteInstance", "DiscardRelationInstance" and "DeleteRelationInstance". We are going to explain each trigger and the use we make of it.

BeforeDeleteInstance

This trigger is activated and an action is performed, before an instance is to be deleted. For our case, this translates in looking, if a node instance is part of the default model or not and take action depending on the situation. This trigger is limited to the instances of nodes and we will use other triggers to look at the instances of relations later on.
One thing that will remain the same for both nodes and relations though, is the way we are going to differentiate between instances of the default model and those that the user has created. To do so, we need to add an attribute to every i* class and relation of the library we are using. We will call this attribute "erasable" and set it to the number "1" by default for instances created by users and hardcode it to "0" during the creation of the instances of the default model, we saw before.

If the trigger is activated, meaning the deletion of a default element has been ordered, we simply need to check if the value of erasable is set to "0" and if it is, we explain to the user through a pop-up box, that this element can't be deleted, because it represents a crucial part of the model. If an instance is deletable, we simply let the system proceed with the deletion of the instance.

DeleteRelationInstance

This trigger is activated if the element that should be deleted, is a relation that is part of the default model. The issue with trying to delete a relation from the default model, is that because of the way ADOxx works, we need to delete and recreate the relation. This means that we need a way to memorise the nodes this relation was connecting, as well as the type of relation it was and later retrieve this information and recreate the relation where it needs to be. The information needed is stored in a record in addition to information about the variant and position the relation needs to be.

This record is stored in the table attached to this model and will be used in the next trigger to restore the relation to where it needs to be. We won't go into the details of the table and how it works and represents the model, since it is complicated and not needed for the understanding of the remaining actions we need to implement.

DiscardRelationInstance

This trigger is activated after the "DeleteRelationInstance" and aims to recreate the relation belonging to the default model after its deletion. This trigger didn't exist when we first implemented this solution and we had to use a number of flags and update functions, which were very dangerous, because they weren't designed to be used in this situation. Since the solution was less than optimal, we collaborated with the development team of the University of Vienna, to add this trigger to the ADOxx framework. We provided the team with information about the way our library was working and its behaviour during deletions and the team updated the ADOxx platform with a newly implemented trigger a few months later.

The way the trigger works is pretty straightforward. We start by retrieving the information about the deleted relation saved in the record within the table of the model and extract one value after another. Those values are then used to recreate the same type of relation than the one that was deleted and position it in the right variant at the right place and linking the same elements of the model.

With these triggers included, we now have a working default model, which can't be deleted, but can be easily completed to represent either SMBSC, VC or BOS in i*.

7 Automated mechanics for instance creations

As we mentioned a few pages ago, we planned on adding functionalities to help the user during the creation of a model. The only functionalities that stood out, were the automated creation of an actor boundary when creating a new role and especially the automated creation of the structure representing the measures in SMBSC.
The addition of boundaries to each created role is easily implemented and requires the use of the "AfterCreateModelingNode" trigger in which we are going to check the type of node created and when it is a role, we get the node's coordinates in the model and variant and we hard-code the initial position of the boundary we want to create near the role. This saves the user a little bit of time and ensures that every role has a boundary. which is a constraint in i*.

For the other operation we want to add, the same trigger is used, but this time we check if the created instance is a goal and if it is used in the SR variant of SMBSC. If this is the case, we create a pop-up window, where we ask the user how many measures are associated to the goal that was just created. The user then chooses an option that is appropriate and we generate the structure resulting from applying the mappings to the user's input. This operation is once again very helpful in creating an i* model representing an SMBSC model and limits the errors the user can make.

To be clear, the elements, that are created this way, aren't part of the default model and thus can be deleted and changed as the user desires, even though we don't recommend them doing so.

CHAPTER 10

Intermediate evaluation for the practical part

In addition to everything we have implemented this far, another major goal was to try and recreate the i* models we created for the theoretical part of this thesis while using our library and check if any i* constraints that were already implemented in the library provided by the University of Vienna are violated while doing so.
We won't go through all the steps of recreating the models we already created once in Visio, since they were explained in the theoretical part of the thesis, we won't show the models or explain how to create them.

The first thing we notice and which we haven't mentioned before, is that no errors came up during the creation of the default models. This means that the mappings used during their creations are correct in regards to the i* constraints. If we continue the creation of the models representing the cases we studied before, we notice once again that no error messages appear.
The fact that these models respect the i* constraints of the library, leads us to our first proof of the validity of the mappings we created. This couldn't be verified by using the Visio models, since, as we said before, the i* stencil of Visio doesn't have any constraints for the placement of the elements.
Of course the fact that no constraints were violated, doesn't in any way mean that our mappings are correct, but only that they work with the current state of the library and for these cases. Nonetheless, this brings us several steps closer to verifying the validity of our mappings and providing a fully functioning library.

Part IV

Conclusion and evaluation

CHAPTER 11

Conclusion

We started the thesis by exposing the questions this thesis would treat and by explaining the different notions and tools, that were going to be used to answer those questions. After those explanations, we looked at creating mappings for SMBSC, BOS and VC and applying them to concrete cases, to have a better understanding of what these mappings would look like and change the different mappings that could cause problems. We finished this part of the thesis with a quick evaluation of the resulting mappings.
Once the three mapping tables looked to be correct, we looked at ways to implement them into ADOxx, using AdoScript and its various triggers. We decided to create automated structures for the more complex mappings and impose default models for the different business strategies to users, so as to facilitate their work. To be consistent with the way we did things for the theoretical part, we recreated the concrete models in the ADOxx environment and looked for any errors linked to the violation of i* constraints. The practical part of the thesis was also finished with a quick evaluation of the work done.

CHAPTER 12

Evaluation

After a first evaluation at the end of chapter 5, where we analysed the linkages between the various notions of the three business strategies and the various elements of i*, we saw first glimpses of the validity of our mappings. We examined the mappings in detail before applying them on real business strategies of three enterprises. In doing so, we managed to have a better view of how our mappings for a particular business strategy would interact between each other and give us the opportunity of adjusting possible conflicting mappings.

After implementing our mappings, through a few automated modelling methods and the creation of the default models, we realized that none of the already implemented basic i* constraints were violated at any point. This, in addition to the fact that after recreating the i* models for the same three real business cases of companies didn't violate any of the i* constraints of the library, comforted us in our assumptions concerning the validity of our findings.

We can also finally answer the question raised during the introduction, about the possibility of representing business strategies through a goal modelling language, with the affirmative. It may not be very intuitive at first, but using the library created in ADOxx, facilitates the creation of those models for the i* goal modelling language.
As a final evaluation of the whole project, we can say that with the creation of three mapping tables, that can be used to represent various business models, and a library that can help users during the creation of their goal models representing business strategies, the project was a success, but also that additional work is still left and that new questions for future works have been raised, which we will discuss in the next chapter.

CHAPTER 13

Future Work

One question that would be interesting to answer, is to see if the newly created library and an older library created by the University of Stockholm, which allowed the creation of UBSMM models (SMBSC, BOS and VC model), could be combined into a single library.
If the result of the first question works, implementing a way to automatically generate the goal model of an UBSMM model that is being created, would be a much for efficient way of using the mappings. The idea behind this is, that while a user is creating his SMBSC model for example, the corresponding i* model is generated in another variant and the user won't need to know or understand the mappings.
Another point that would require additional work, is the implementation of all the i* constraints. This hasn't been done until now, since there are many actions that are considered as not recommended, but which could be imposed as constraints in the library. Doing so, would most likely require an in depth rework of the mappings, since those constraints could only be examined by hand until now.

Bibliography

[1] Heinz Ahn. Applying the balanced scorecard concept: An experience report. *Elsevier Science Ltd.*, 2001.

[2] Eric-Oluf Svee Constantinos Giannoulis and Jelena Zdravkovic. Capturing consumer preference in system requirements through business strategy. Technical report, Stockholm University, 2013.

[3] Michaël Petit Constantinos Giannoulis and Jelena Zdravkovic. Modeling competition-driven business strategy for business it alignment. Technical report, 2011.

[4] Constantinos Giannoulis. Modeling business strategy, for business-it alignment. Technical report, May 2011.

[5] Constantinos Giannoulis. Schema integration process for ubsmm. Technical report, Stockholm University, 2011.

[6] Constantinos Giannoulis. *Model-driven Alignment: Linking Business Strategy with Information Systems*. PhD thesis, Stockholm University, May 2014.

[7] Constantinos Giannoulis and Jelena Zdravkovic. Modeling strategy maps and balanced scorecards using *i**. In *iStar*, pages 90–95, 2011.

[8] Constantinos Giannoulis and Jelena Zdravkovic. Linking strategic innovation to requirements: a look into blue ocean strategy. In *PoEM (Short Papers)*, 2012.

[9] Constantinos Giannoulis, Iyad Zikra, Maria Bergholtz, Jelena Zdravkovic, Janis Stirna, and Paul Johannesson. A comparative analysis of enterprise modeling approaches for modeling business strategy. In *PoEM (Short Papers)*, pages 193–204, 2013.

[10] Robert S. Kaplan and David P. Norton. *The strategy map: guide to aligning intangible assets*. Emerald Group Publishing Limited, 2004.

[11] Margit Schwab. The *i** method: Conceptualization concept & implementation procedure for adoxx. Technical report, Universität Wien, June 2011.

[12] Charles B. Stabell and Øystein D. Fjeldstad. Configuring value for competitive advantage:on chains, shops, and networks. *Strategic management journal*, 1998.

Appendices

Detailed view of the SMBSC model for ABB

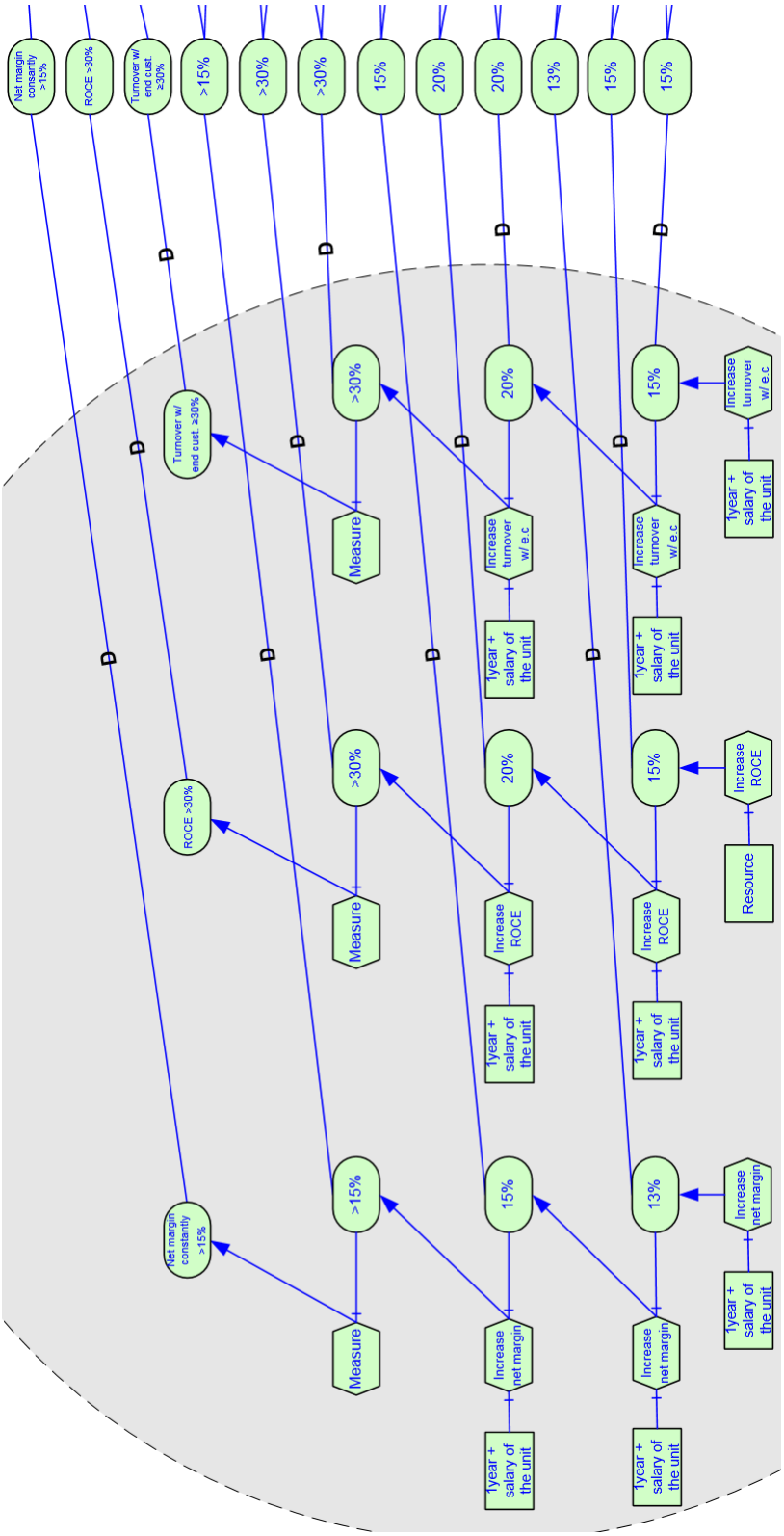

Figure 1: Mapping of the financial perspective of ABB in i* using Visio

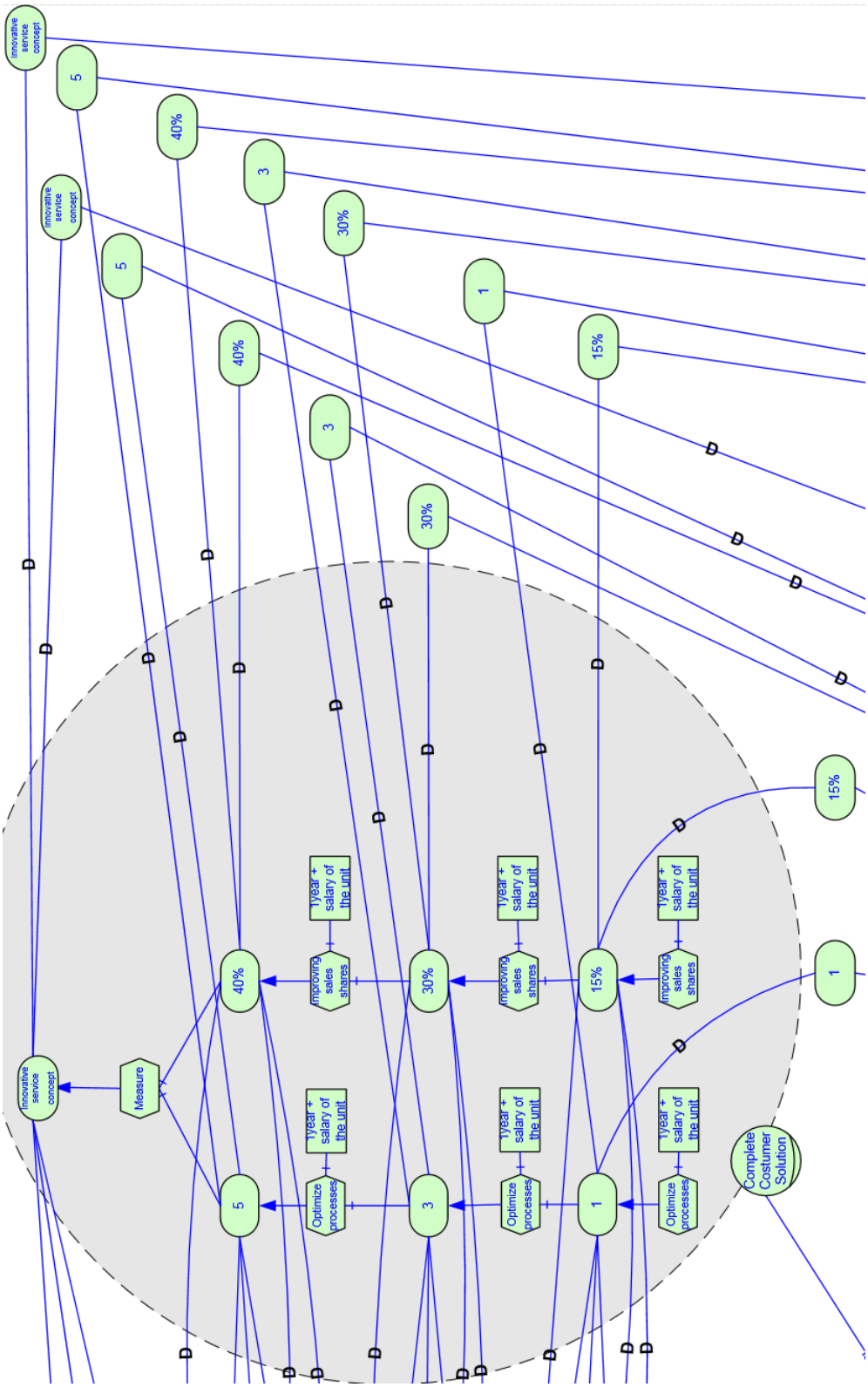

Figure 2: Mapping of the customer perspective of ABB in i* using Visio

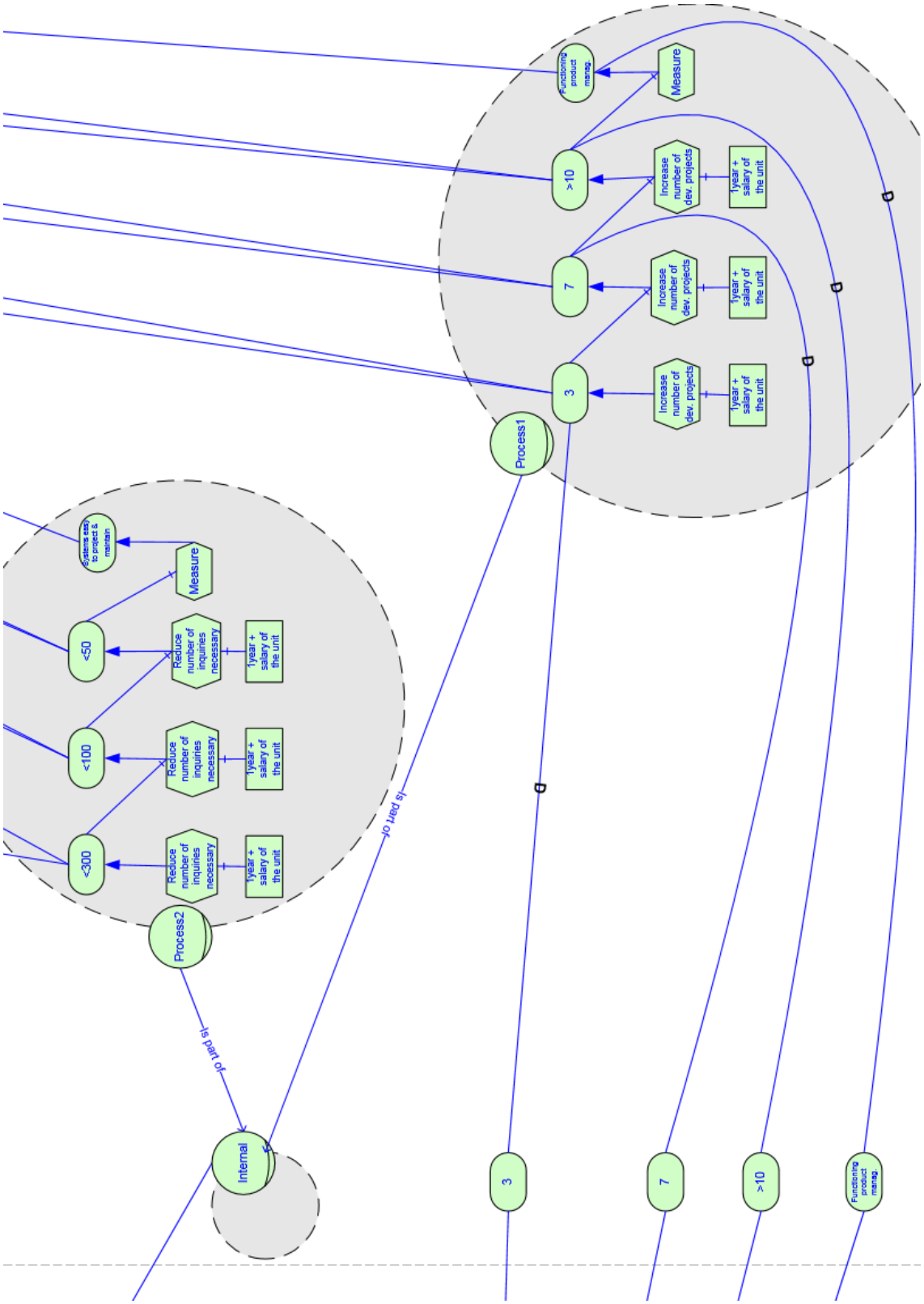

Figure 3: Mapping of the internal perspective of ABB in i* using Visio

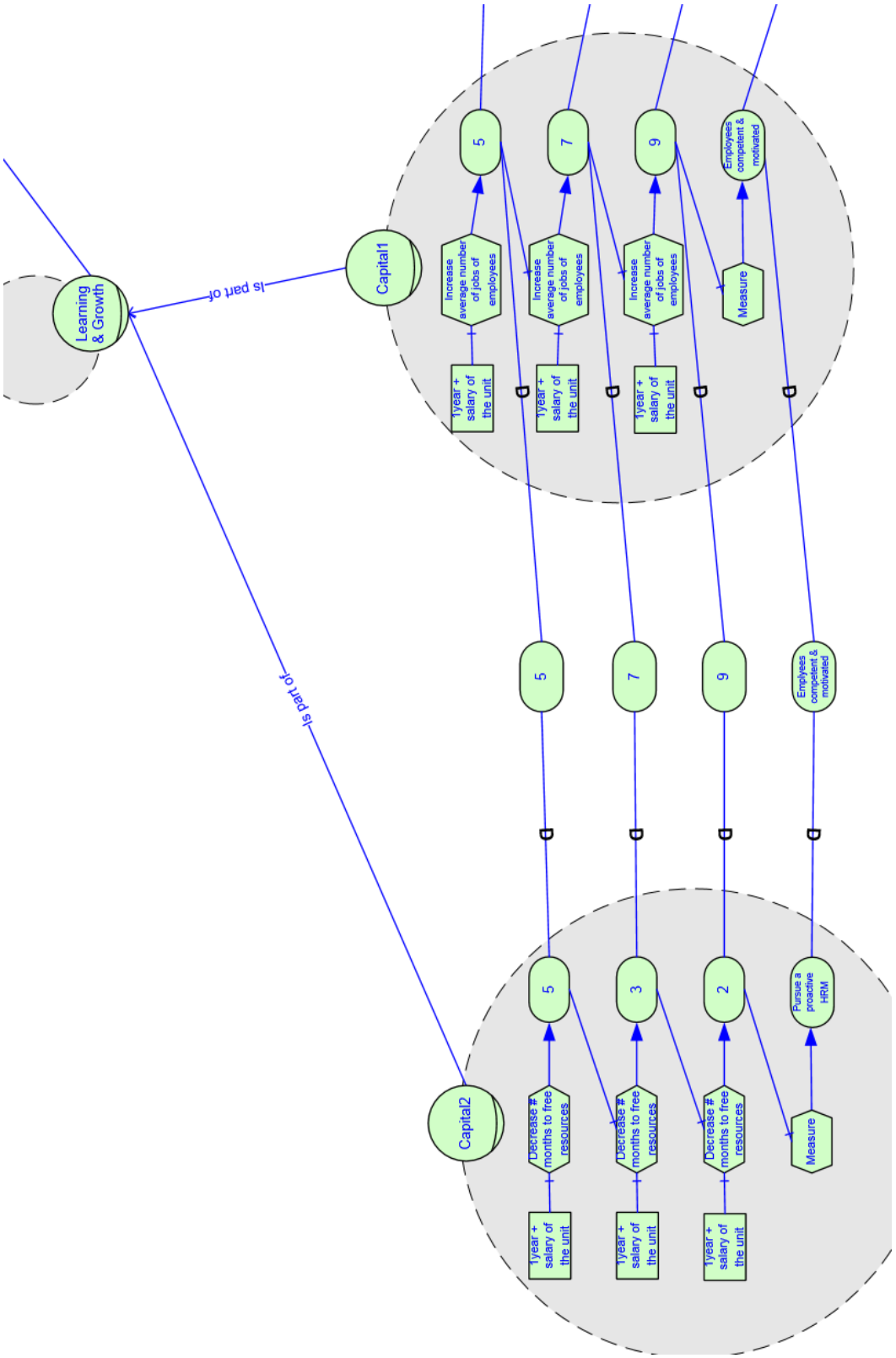

Figure 4: Mapping of the feature perspective of ABB in i* using Visio

AdoScript in the External Coupling

```
1
2  ON_EVENT "AfterCreateModelingConnector" {
3    SETG class_id: (classid)
4    SETG obj_id: (objid)
5    SETG nCreatedModelID:(modelid)
6    CC "AdoScript" FREAD file:     ("db:\\createConnector.asc")
7    IF (text = ""  OR  ecode != 0) {
8      CC "AdoScript" ERRORBOX ("Error loading the CreateConnector file")
9    }
10   ELSE {
11     EXECUTE (text)
12   }
13 }
14
15 ON_EVENT "AfterCreateModelingNode" {
16   SETG class_id: (classid)
17   SETG obj_id: (objid)
18   SETG nCreatedModelID:(modelid)
19   CC "AdoScript" FREAD file:     ("db:\\createInstance.asc")
20   IF (text = ""  OR  ecode != 0) {
21    CC "AdoScript" ERRORBOX ("Error loading the CreateInstance file")
22   }
23   ELSE {
24    EXECUTE (text)
25   }
26
27   CC "Core" GET_CLASS_NAME classid: (class_id)
28   SETG fromclassname:(classname)
29
30   CC "Core" GET_ATTR_ID classid:(class_id) attrname:"Position"
31   SETG attr_id: (attrid)
32
33   CC "Core" GET_CLASS_ID classname:"Task"
34   SETG nCreateClassTaskID:(classid)
35
36   CC "Core" GET_CLASS_ID classname:"Goal"
37   SETG nCreateClassGoalID:(classid)
38
39   CC "Core" GET_CLASS_ID classname:"Means-end Link"
```

```
40   SETG nMeansEndID:(classid)
41
42   CC "Core" GET_CLASS_ID classname:"Resource"
43   SETG nCreateClassResourceID:(classid)
44
45   CC "Core" GET_CLASS_ID classname:"Decomposition Link"
46   SETG nDecompID:(classid)
47
48   CC "Core" GET_CLASS_ID classname:"Boundary"
49   SETG nCreateClassBoundaryID:(classid)
50
51   IF (fromclassname = "Goal") {
52     IF (origin=0) {
53       CC "AdoScript" FREAD file:    ("db:\\AfterCreateGoal2.asc")
54       IF (text = ""  OR  ecode != 0) {
55         CC "AdoScript" ERRORBOX ("Error loading the AfterCreateGoal(SM) file")
56       }
57     ELSE {
58         EXECUTE (text)
59       }
60     }
61   }
62   IF (fromclassname = "Role") {
63     IF (origin=0) {
64       CC "AdoScript" FREAD file:    ("db:\\AfterCreateRole.asc")
65       IF (text = ""  OR  ecode != 0) {
66         CC "AdoScript" ERRORBOX ("Error loading the AfterCreateRole file")
67       }
68     ELSE {
69         EXECUTE (text)
70       }
71     }
72   }
73 }
74
75 ON_EVENT "AfterCreateModelWindow" {
76   SETG nCreatedModelID:(modelid)
77   CC "AdoScript" FREAD file:    ("db:\\latest_default.asc")
78   IF (text = ""  OR  ecode != 0) {
79     CC "AdoScript" ERRORBOX ("Error loading the MainDefaultModel file")
80   }
81   ELSE {
82     EXECUTE (text)
83   }
84 }
85
86 ON_EVENT "BeforeDeleteInstance" {
87   EVENT_LOG msgType:"EVENT_LOG" message:("BeforeDeleteInstance")
88   CC "Core" GET_CLASS_NAME classid:(classid)
```

```
89   # This is needes since otherwise any delete of a record row = instance is run
       through here as well
90   IF (classname <> "Default Relations") {
91     SETL nDeleteInstanceID:(instid)
92     # available parameters are instid, classid, modelid
93     # check if object is erasable
94     CC "Core" GET_ATTR_VAL objid:(nDeleteInstanceID) attrname:("Erasable")
95     SETL bErasable:(val)
96     IF (bErasable = 0) {
97       CC "AdoScript" INFOBOX "This object is part of the default model and
       cannot be deleted."
98       EXIT -1
99     }
100  }
101 }
102
103 ON_EVENT "DiscardRelationInstance" {
104   CC "Modeling" GET_ACT_MODEL
105   SETL nCreatedModelID:(modelid)
106   IF (nCreatedModelID != -1) {
107     EVENT_LOG msgType:"EVENT_LOG" message:("DiscardRelationInstance")
108     CC "Core" GET_ATTR_ID classid:bp-model attrname:"Default Relations"
109     SETL nRecAttrID:(attrid)
110     CC "Core" GET_ALL_REC_ATTR_ROW_IDS objid:(nCreatedModelID) attrid:(
       nRecAttrID)
111     SETL lRowIDs:(rowids)
112     FOR sRowID in:(lRowIDs) {
113       CC "Core" GET_ATTR_VAL objid:(VAL sRowID) attrname:("StartNode")
114       SETL nFromInstanceID:(VAL val)
115       CC "Core" GET_ATTR_VAL objid:(VAL sRowID) attrname:("EndNode")
116       SETL nToInstanceID:(VAL val)
117       CC "Core" GET_ATTR_VAL objid:(VAL sRowID) attrname:("RelationClass")
118       SETL nRelationClassID:(VAL val)
119       CC "Core" GET_ATTR_VAL objid:(VAL sRowID) attrname:("Positions")
120       SETL sPositions:(val)
121       CC "Core" GET_ATTR_VAL objid:(VAL sRowID) attrname:("Variants")
122       SETL sVariants:(val)
123       CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
       nFromInstanceID) toobjid:(nToInstanceID) classid:(nRelationClassID)
124       SETL nRelationCreated:(objid)
125       CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions")
       val:(sPositions)
126       CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("___Variants___")
        val:(sVariants)
127       CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val
       :0
128       CC "Core" REMOVE_REC_ROW objid:(nCreatedModelID) attrid:(nRecAttrID)
       rowid:(VAL sRowID)
129     }
130   }
```

```
131 }
132
133 ON_EVENT "DeleteRelationInstance" {
134   EVENT_LOG msgType:"EVENT_LOG" message:("DeleteRelationInstance")
135   CC "Core" GET_MODEL_ID objid:(relationinstanceid)
136   SETL nCreatedModelID:(modelid)
137   CC "Core" GET_CLASS_NAME classid: (relationclassid)
138   SETL relationClassName: (classname)
139   IF (relationClassName <> "Is inside") {
140     # check if object is erasable
141     CC "Core" GET_ATTR_VAL objid:(relationinstanceid) attrname:("Erasable")
142     SETL bErasable:(val)
143     IF (bErasable = 0) {
144       CC "Core" GET_ATTR_VAL objid:(relationinstanceid) attrname:"Positions"
145       SETG sPositionsVal:(val)
146       CC "Core" GET_ATTR_VAL objid:(relationinstanceid) attrname:"__Variants__"
147       SETG sVariantsVal:(val)
148       CC "Core" GET_ATTR_ID classid:bp-model attrname:"Default Relations"
149       CC "Core" ADD_REC_ROW objid:(nCreatedModelID) attrid:(attrid)
150       SETL nRelationRowID:(rowid)
151       CC "Core" SET_ATTR_VAL objid:(nRelationRowID) attrname:("StartNode") val
    :(STR frominstanceid)
152       CC "Core" SET_ATTR_VAL objid:(nRelationRowID) attrname:("EndNode") val:(
    STR toinstanceid)
153       CC "Core" SET_ATTR_VAL objid:(nRelationRowID) attrname:("Positions") val
    :(sPositionsVal)
154       CC "Core" SET_ATTR_VAL objid:(nRelationRowID) attrname:("RelationClass")
    val:(STR relationclassid)
155       CC "Core" SET_ATTR_VAL objid:(nRelationRowID) attrname:("Variants") val:(
    sVariantsVal)
156     CC "AdoScript" INFOBOX "This relation is part of the default model and
    cannot be deleted."
157     }
158   }
159 }
```

AdoScript for the BOS default models

We will only show the code for the default models for the SD and SR models of BOS, since the default models for SMBSC and VC can easily be deduced from this code.

```
1
2  ############################################################
3  ########################## SD_BOS ##########################
4  ############################################################
5
6  # set current variant to "Strategic Dependency Model for BO"
7  CC "Drawing" SET_ACTIVE_VARIANT modelid:(nCreatedModelID) variant:"Strategic
     Dependency Model for BO"
8  EVENT_LOG msgType:"EVENT_LOG" message:("Model is initializing now for variant
       Strategic Dependency Model for BO")
9
10 # MAIN ROLE BO
11 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nCreateClassID)
     objname:"Main_Actor_BO1"
12 SETL nCreatedRole1ObjID:(objid)
13 CC "Core" SET_ATTR_VAL objid:(nCreatedRole1ObjID) attrname:("Position") val:"
     NODE x:x:118.5cm y:119cm w:2cm h:1.2cm index:1 iuv:0"
14 CC "Core" SET_ATTR_VAL objid:(nCreatedRole1ObjID) attrname:("Erasable") val:0
15
16 # BOUNDARY
17 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(
     nBoundaryCreateClassID) objname:"Boundary_Main_Role1"
18 SETL nCreatedBoundaryObjID1:(objid)
19 CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Position")
     val:"NODE x:118.5cm y:116cm w:6cm h:6cm index:1 iuv:0"
20 CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Erasable")
     val:0
21
22
23 # ROLE BO
24 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nCreateClassID)
     objname:"Role_BO_1"
25 SETL nCreatedRole2ObjID:(objid)
26 CC "Core" SET_ATTR_VAL objid:(nCreatedRole2ObjID) attrname:("Position") val:"
     NODE x:110cm y:125cm w:2cm h:1.2cm index:1 iuv:0"
27 CC "Core" SET_ATTR_VAL objid:(nCreatedRole2ObjID) attrname:("Erasable") val:0
28
```

```
29  # BOUNDARY
30  CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(
      nBoundaryCreateClassID) objname:"Boundary_Role_BO_1"
31  SETL nCreatedBoundaryObjID1:(objid)
32  CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Position")
      val:"NODE x:108.5cm y:128cm w:6cm h:6cm index:1 iuv:0"
33  CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Erasable")
      val:0
34
35
36  # ROLE BO
37  CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nCreateClassID)
      objname:"Role_BO_2"
38  SETL nCreatedRole3ObjID:(objid)
39  CC "Core" SET_ATTR_VAL objid:(nCreatedRole3ObjID) attrname:("Position") val:"
      NODE x:127cm y:125cm w:2cm h:1.2cm index:1 iuv:0"
40  CC "Core" SET_ATTR_VAL objid:(nCreatedRole3ObjID) attrname:("Erasable") val:0
41
42  # BOUNDARY
43  CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(
      nBoundaryCreateClassID) objname:"Boundary_Role_BO_2"
44  SETL nCreatedBoundaryObjID1:(objid)
45  CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Position")
      val:"NODE x:128.5cm y:128cm w:6cm h:6cm index:1 iuv:0"
46  CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Erasable")
      val:0
47
48  # 4 goals as dependums
49  CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Be divergent1"
50  SETL nCreatedGoalSR1ObjID:(objid)
51  CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR1ObjID) attrname:("Position") val
      :"NODE x:112cm y:122cm w:2cm h:1.2cm index:1 iuv:0"
52  CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR1ObjID) attrname:("Erasable") val
      :0
53
54  CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Be focused1"
55  SETL nCreatedGoalSR2ObjID:(objid)
56  CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR2ObjID) attrname:("Position") val
      :"NODE x:116cm y:122cm w:2cm h:1.2cm index:1 iuv:0"
57  CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR2ObjID) attrname:("Erasable") val
      :0
58
59  CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Be divergent2"
60  SETL nCreatedGoalSR3ObjID:(objid)
61  CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR3ObjID) attrname:("Position") val
      :"NODE x:120cm y:122cm w:2cm h:1.2cm index:1 iuv:0"
```

```
62  CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR3ObjID) attrname:("Erasable") val
      :0
63
64  CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Be focused2"
65  SETL nCreatedGoalSR4ObjID:(objid)
66  CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR4ObjID) attrname:("Position") val
      :"NODE x:124cm y:122cm w:2cm h:1.2cm index:1 iuv:0"
67  CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR4ObjID) attrname:("Erasable") val
      :0
68
69
70  # CREATION OF DEPENDENCY LINKS
71  CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedRole1ObjID) toobjid:(nCreatedGoalSR1ObjID) classid:(
      nDependencyCreateClassID)
72  SETL nRelationCreated:(objid)
73  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
74  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
75
76  CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedRole1ObjID) toobjid:(nCreatedGoalSR2ObjID) classid:(
      nDependencyCreateClassID)
77  SETL nRelationCreated:(objid)
78  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
79  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
80
81  CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedRole1ObjID) toobjid:(nCreatedGoalSR3ObjID) classid:(
      nDependencyCreateClassID)
82  SETL nRelationCreated:(objid)
83  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
84  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
85
86  CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedRole1ObjID) toobjid:(nCreatedGoalSR4ObjID) classid:(
      nDependencyCreateClassID)
87  SETL nRelationCreated:(objid)
88  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
89  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
90
91  CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalSR1ObjID) toobjid:(nCreatedRole2ObjID) classid:(
      nDependencyCreateClassID)
92  SETL nRelationCreated:(objid)
```

```
93  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
94  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
95
96  CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalSR2ObjID) toobjid:(nCreatedRole2ObjID) classid:(
      nDependencyCreateClassID)
97  SETL nRelationCreated:(objid)
98  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
99  CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
100
101 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalSR3ObjID) toobjid:(nCreatedRole3ObjID) classid:(
      nDependencyCreateClassID)
102 SETL nRelationCreated:(objid)
103 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
104 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
105
106 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalSR4ObjID) toobjid:(nCreatedRole3ObjID) classid:(
      nDependencyCreateClassID)
107 SETL nRelationCreated:(objid)
108 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
109 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
110
111 ##################################################################
112 ########################## SR_BOS ################################
113 ##################################################################
114
115 # set current variant to "Strategic Rationale Model for BO"
116 CC "Drawing" SET_ACTIVE_VARIANT modelid:(nCreatedModelID) variant:"Strategic
      Rationale Model for BO"
117 EVENT_LOG msgType:"EVENT_LOG" message:("Model is initializing now for variant
      Strategic Rationale Model for BO")
118
119
120 # MAIN ROLE BO
121 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nCreateClassID)
      objname:"Main_Actor_BO"
122 SETL nCreatedSR1ObjID:(objid)
123 CC "Core" SET_ATTR_VAL objid:(nCreatedSR1ObjID) attrname:("Position") val:"
      NODE x:120cm y:115cm w:2cm h:1.2cm index:1 iuv:0"
124 CC "Core" SET_ATTR_VAL objid:(nCreatedSR1ObjID) attrname:("Erasable") val:0
125
126 # BOUNDARY
127 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(
      nBoundaryCreateClassID) objname:"Boundary_Main_Role"
```

```
128 SETL nCreatedBoundaryObjID1:(objid)
129 CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Position")
      val:"NODE x:118.5cm y:118cm w:6cm h:6cm index:1 iuv:0"
130 CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Erasable")
      val:0
131
132 # MAIN ROLE GOAL & TASK
133 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nTaskCreateClassID)
      objname:"Main_Role_Task_BO1"
134 SETL nCreatedStartObj1ID:(objid)
135 CC "Core" SET_ATTR_VAL objid:(nCreatedStartObj1ID) attrname:("Position") val:
      "NODE x:118.5cm y:117cm w:2cm h:1.2cm index:1 iuv:0"
136 CC "Core" SET_ATTR_VAL objid:(nCreatedStartObj1ID) attrname:("Erasable") val
      :0
137
138 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Main_Role_Goal_BO1"
139 SETL nCreatedGoalBound1ObjID:(objid)
140 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalBound1ObjID) attrname:("Position")
      val:"NODE x:118.5cm y:120cm w:2cm h:1.2cm index:1 iuv:0"
141 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalBound1ObjID) attrname:("Erasable")
      val:0
142
143 # Create a relation that connects both elements
144 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedStartObj1ID) toobjid:(nCreatedGoalBound1ObjID) classid:(
      nMeansCreateClassID)
145 SETL nRelationCreated:(objid)
146 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
147 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
148
149
150 # ROLE BO 2
151 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nCreateClassID)
      objname:"Role_BO_2"
152 SETL nCreatedSR2ObjID:(objid)
153 CC "Core" SET_ATTR_VAL objid:(nCreatedSR2ObjID) attrname:("Position") val:"
      NODE x:110cm y:125cm w:2cm h:1.2cm index:1 iuv:0"
154 CC "Core" SET_ATTR_VAL objid:(nCreatedSR2ObjID) attrname:("Erasable") val:0
155
156 # BOUNDARY
157 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(
      nBoundaryCreateClassID) objname:"Boundary_Role_BO2"
158 SETL nCreatedBoundaryObjID1:(objid)
159 CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Position")
      val:"NODE x:108.5cm y:128cm w:6cm h:6cm index:1 iuv:0"
160 CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Erasable")
      val:0
161
```

```
# MAIN ROLE GOAL & TASK
CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nTaskCreateClassID) objname:"Task_BO2"
SETL nCreatedTask2ObjID:(objid)
CC "Core" SET_ATTR_VAL objid:(nCreatedTask2ObjID) attrname:("Position") val:"NODE x:109cm y:129.5cm w:2cm h:1.2cm index:1 iuv:0"
CC "Core" SET_ATTR_VAL objid:(nCreatedTask2ObjID) attrname:("Erasable") val:0

CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID) objname:"Goal_BO2"
SETL nCreatedGoalBound2ObjID:(objid)
CC "Core" SET_ATTR_VAL objid:(nCreatedGoalBound2ObjID) attrname:("Position") val:"NODE x:109cm y:127cm w:2cm h:1.2cm index:1 iuv:0"
CC "Core" SET_ATTR_VAL objid:(nCreatedGoalBound2ObjID) attrname:("Erasable") val:0

# Create a relation that connects both elements
CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(nCreatedTask2ObjID) toobjid:(nCreatedGoalBound2ObjID) classid:(nMeansCreateClassID)
SETL nRelationCreated:(objid)
CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"EDGE 0 index:1 iuv:0"
CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0


# ROLE BO 3
CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nCreateClassID) objname:"Role_BO_3"
SETL nCreatedSR3ObjID:(objid)
CC "Core" SET_ATTR_VAL objid:(nCreatedSR3ObjID) attrname:("Position") val:"NODE x:130cm y:125cm w:2cm h:1.2cm index:1 iuv:0"
CC "Core" SET_ATTR_VAL objid:(nCreatedSR3ObjID) attrname:("Erasable") val:0

# BOUNDARY
CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nBoundaryCreateClassID) objname:"Boundary_Role_BO3"
SETL nCreatedBoundaryObjID1:(objid)
CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Position") val:"NODE x:128.5cm y:128cm w:6cm h:6cm index:1 iuv:0"
CC "Core" SET_ATTR_VAL objid:(nCreatedBoundaryObjID1) attrname:("Erasable") val:0

# MAIN ROLE GOAL & TASK
CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nTaskCreateClassID) objname:"Task_BO3"
SETL nCreatedTask3ObjID:(objid)
CC "Core" SET_ATTR_VAL objid:(nCreatedTask3ObjID) attrname:("Position") val:"NODE x:128cm y:129.5cm w:2cm h:1.2cm index:1 iuv:0"
CC "Core" SET_ATTR_VAL objid:(nCreatedTask3ObjID) attrname:("Erasable") val:0
```

```
197
198 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Goal_BO3"
199 SETL nCreatedGoalBound3ObjID:(objid)
200 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalBound3ObjID) attrname:("Position")
      val:"NODE x:128cm y:127cm w:2cm h:1.2cm index:1 iuv:0"
201 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalBound3ObjID) attrname:("Erasable")
      val:0
202
203 # Create a relation that connects both elements
204 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedTask3ObjID) toobjid:(nCreatedGoalBound3ObjID) classid:(
      nMeansCreateClassID)
205 SETL nRelationCreated:(objid)
206 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
207 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
208
209
210 # CREATION OF 4 GOALS
211 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Be divergent1"
212 SETL nCreatedGoalSR1ObjID:(objid)
213 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR1ObjID) attrname:("Position") val
      :"NODE x:113cm y:123cm w:2cm h:1.2cm index:1 iuv:0"
214 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR1ObjID) attrname:("Erasable") val
      :0
215
216 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Be focused1"
217 SETL nCreatedGoalSR2ObjID:(objid)
218 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR2ObjID) attrname:("Position") val
      :"NODE x:116cm y:123cm w:2cm h:1.2cm index:1 iuv:0"
219 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR2ObjID) attrname:("Erasable") val
      :0
220
221 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Be divergent2"
222 SETL nCreatedGoalSR3ObjID:(objid)
223 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR3ObjID) attrname:("Position") val
      :"NODE x:119cm y:123cm w:2cm h:1.2cm index:1 iuv:0"
224 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR3ObjID) attrname:("Erasable") val
      :0
225
226 CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nGoalCreateClassID)
      objname:"Be focused2"
227 SETL nCreatedGoalSR4ObjID:(objid)
228 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR4ObjID) attrname:("Position") val
      :"NODE x:122cm y:123cm w:2cm h:1.2cm index:1 iuv:0"
```

```
229 CC "Core" SET_ATTR_VAL objid:(nCreatedGoalSR4ObjID) attrname:("Erasable") val
      :0
230
231
232  # CREATION OF DEPENDENCY LINKS
233 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalBound1ObjID) toobjid:(nCreatedGoalSR1ObjID) classid:(
      nDependencyCreateClassID)
234 SETL nRelationCreated:(objid)
235 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
236 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
237
238 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalBound1ObjID) toobjid:(nCreatedGoalSR2ObjID) classid:(
      nDependencyCreateClassID)
239 SETL nRelationCreated:(objid)
240 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
241 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
242
243 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalBound1ObjID) toobjid:(nCreatedGoalSR3ObjID) classid:(
      nDependencyCreateClassID)
244 SETL nRelationCreated:(objid)
245 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
246 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
247
248 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalBound1ObjID) toobjid:(nCreatedGoalSR4ObjID) classid:(
      nDependencyCreateClassID)
249 SETL nRelationCreated:(objid)
250 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
251 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
252
253 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalSR1ObjID) toobjid:(nCreatedGoalBound2ObjID) classid:(
      nDependencyCreateClassID)
254 SETL nRelationCreated:(objid)
255 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
256 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
257
258 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalSR2ObjID) toobjid:(nCreatedGoalBound2ObjID) classid:(
      nDependencyCreateClassID)
259
260 SETL nRelationCreated:(objid)
```

```
261 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
262 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
263
264 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalSR3ObjID) toobjid:(nCreatedGoalBound3ObjID) classid:(
      nDependencyCreateClassID)
265 SETL nRelationCreated:(objid)
266 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
267 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
268
269 CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
      nCreatedGoalSR4ObjID) toobjid:(nCreatedGoalBound3ObjID) classid:(
      nDependencyCreateClassID)
270 SETL nRelationCreated:(objid)
271 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:"
      EDGE 0 index:1 iuv:0"
272 CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Erasable") val:0
```

Code for the automated creation for SMBSC

```
1
2  SET nMainGoalObjID:(obj_id)
3  SET nMainGoalID:(class_id)
4  SET nGoalAttrID:(attr_id)
5  SET nClassName:(fromclassname)
6  CC "Drawing" GET_ACTIVE_VARIANT modelid:(nCreatedModelID)
7  SET activeVariant: (variant)
8
9  IF (nClassName = "Goal")
10 {
11   IF (activeVariant = "Strategic Rationale Model for SMBSC")
12   {
13   CC "AdoScript" LISTBOX
14   boxtext:"Please choose the number of measures this objective has!"
15   entries:"1;2;3;4;5;6;7;8;9;10" toksep:";"
16   title:"Measures for an objective!" oktext:"Click me!" boxtext:"Choose your
       entry:" selection:"1"
17
18   IF (endbutton = "ok")
19   {
20     SET measure_counter: 0
21     SET int_selection: (valarray(selection)[0])
22     SET selection_array: ({0,1,2,3,4,5,6,7,8,9,10})
23
24     FOR i from: 0 to: (int_selection)
25     {
26       SET selection_array[i]: (i+1)
27     }
28
29     SET nNameString: (strarray(tokstr(selection_array))[measure_counter])
30     SET obj_array: ({obj_id})
31     SET nObjString: (strarray(tokstr(obj_array))[0])
32
33     # create the task for the Measure
34     CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(nCreateClassTaskID)
        objname:("Task"+nObjString)
35     SET nTaskID:(objid)
36
37     CC "Core" GET_ATTR_ID classid:(nMainGoalID) attrname:"Position"
```

```
38      CC "Core" GET_ATTR_VAL objid:(nMainGoalObjID) attrid:(nGoalAttrID)
39
40      LEO parse: (val) get-tmm-value:x:"x" get-tmm-value:y:"y"
41
42      SET x:(x)
43      SET y:(y + 3cm)
44      CC "Modeling" SET_OBJ_POS objid: (nTaskID) x:(x) y:(y)
45
46        CC "Core" GET_ATTR_ID classid:(nCreateClassTaskID) attrname:"Position"
47      SETL nPositionAttrID:(attrid)
48      CC "Core" GET_ATTR_VAL objid:(nTaskID) attrid:(nPositionAttrID)
49      SETL nPositionTaskVal: (val)
50      CC "Core" SET_ATTR_VAL objid:(nTaskID) attrname:("Position") val:(
        nPositionTaskVal + " iuv:0")
51
52      # Create a relation that connects both elements
53      CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(nTaskID)
        toobjid:(nMainGoalObjID) classid:(nMeansEndID)
54      SETL nRelationCreated: (objid)
55      CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:
        "EDGE 0 index:1 iuv:0"
56          CC "Core" GET_ATTR_ID classid:(nMeansEndID) attrname:"Positions"
57        SETL nPositionsAttrID:(attrid)
58        CC "Core" GET_ATTR_VAL objid:(nRelationCreated) attrid:(nPositionsAttrID)
59        SETL nPositionsVal: (val)
60      EVENT_LOG msgType:"EVENT_LOG" message:(nPositionsVal)
61
62          # create a resource
63      CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(
        nCreateClassResourceID) objname:("Resource"+nObjString)
64      SET nResourceID:(objid)
65
66      CC "Core" GET_ATTR_ID classid:(nMainGoalID) attrname:"Position"
67      CC "Core" GET_ATTR_VAL objid:(nMainGoalObjID) attrid:(nGoalAttrID)
68
69      LEO parse: (val) get-tmm-value:x:"x" get-tmm-value:y:"y"
70
71      SET x:(x + 3cm )
72      SET y:(y + 3cm)
73      CC "Modeling" SET_OBJ_POS objid: (nResourceID) x:(x) y:(y)
74
75      CC "Core" GET_ATTR_ID classid:(nCreateClassResourceID) attrname:"Position"
76      SETL nPositionAttrID:(attrid)
77      CC "Core" GET_ATTR_VAL objid:(nResourceID) attrid:(nPositionAttrID)
78      SETL nPositionResVal: (val)
79      CC "Core" SET_ATTR_VAL objid:(nResourceID) attrname:("Position") val:(
        nPositionResVal + " iuv:0")
80
81      # Create a relation that connects both elements
```

```
82      CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(nResourceID
        ) toobjid:(nTaskID) classid:(nDecompID)
83      SETL nRelationCreated: (objid)
84      CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions") val:
        "EDGE 0 index:1 iuv:0"
85
86      WHILE (measure_counter < int_selection)
87      {
88        SET nNameString: (strarray(tokstr(selection_array))[measure_counter])
89        SET obj_array: ({obj_id})
90        SET nObjString: (strarray(tokstr(obj_array))[0])
91
92        # create a goal
93        CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(
        nCreateClassGoalID) objname:("Milestone"+nNameString+nObjString)
94        SET measure_counter: (measure_counter+1)
95        SET nGoalID1:(objid)
96
97        CC "Core" GET_ATTR_ID classid:(nMainGoalID) attrname:"Position"
98        CC "Core" GET_ATTR_VAL objid:(nMainGoalObjID) attrid:(nGoalAttrID)
99
100       LEO parse: (val) get-tmm-value:x:"x" get-tmm-value:y:"y"
101       SET x_coord: ((measure_counter)*3)
102       SET x1: (CM x_coord)
103
104       SET x:(x -6cm + x1)
105       SET y:(y + 6cm)
106       CC "Modeling" SET_OBJ_POS objid: (nGoalID1) x:(x) y:(y)
107
108         CC "Core" GET_ATTR_ID classid:(nCreateClassGoalID) attrname:"Position"
109       SETL nPositionAttrID:(attrid)
110       CC "Core" GET_ATTR_VAL objid:(nGoalID1) attrid:(nPositionAttrID)
111       SETL nPositionGoalVal: (val)
112       CC "Core" SET_ATTR_VAL objid:(nGoalID1) attrname:("Position") val:(
        nPositionGoalVal + " iuv:0")
113
114       # Create a relation that connects both elements
115       CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(nGoalID1)
         toobjid:(nTaskID) classid:(nDecompID)
116       SETL nRelationCreated: (objid)
117       CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions")
        val:"EDGE 0 index:1 iuv:0"
118
119       # create a task
120       CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(
        nCreateClassTaskID) objname:("Task"+nNameString+nObjString)
121       SET nTaskID1:(objid)
122
123       CC "Core" GET_ATTR_ID classid:(nMainGoalID) attrname:"Position"
124       CC "Core" GET_ATTR_VAL objid:(nMainGoalObjID) attrid:(nGoalAttrID)
```

```
125
126     LEO parse: (val) get-tmm-value:x:"x" get-tmm-value:y:"y"
127     SET x_coord: ((measure_counter)*3)
128     SET x1: (CM x_coord)
129
130     SET x:(x -6cm + x1)
131     SET y:(y + 9cm)
132     CC "Modeling" SET_OBJ_POS objid: (nTaskID1) x:(x) y:(y)
133
134       CC "Core" GET_ATTR_ID classid:(nCreateClassTaskID) attrname:"Position"
135     SETL nPositionAttrID:(attrid)
136     CC "Core" GET_ATTR_VAL objid:(nTaskID1) attrid:(nPositionAttrID)
137     SETL nPositionTaskVal: (val)
138     CC "Core" SET_ATTR_VAL objid:(nTaskID1) attrname:("Position") val:(
    nPositionTaskVal + " iuv:0")
139
140     # Create a relation that connects both elements
141     CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(nTaskID1)
     toobjid:(nGoalID1) classid:(nMeansEndID)
142     SETL nRelationCreated: (objid)
143     CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions")
    val:"EDGE 0 index:1 iuv:0"
144
145           # create a ressource
146     CC "Core" CREATE_OBJ modelid:(nCreatedModelID) classid:(
    nCreateClassResourceID) objname:("Resource"+nNameString+nObjString)
147     SET nResourceID1:(objid)
148
149     CC "Core" GET_ATTR_ID classid:(nMainGoalID) attrname:"Position"
150     CC "Core" GET_ATTR_VAL objid:(nMainGoalObjID) attrid:(nGoalAttrID)
151
152     LEO parse: (val) get-tmm-value:x:"x" get-tmm-value:y:"y"
153     SET x_coord: ((measure_counter)*3)
154     SET x1: (CM x_coord)
155
156     SET x:(x -6cm + x1)
157     SET y:(y + 11cm)
158     CC "Modeling" SET_OBJ_POS objid: (nResourceID1) x:(x) y:(y)
159
160     CC "Core" GET_ATTR_ID classid:(nCreateClassResourceID) attrname:"Position
    "
161     SETL nPositionAttrID:(attrid)
162     CC "Core" GET_ATTR_VAL objid:(nResourceID1) attrid:(nPositionAttrID)
163     SETL nPositionResVal: (val)
164     CC "Core" SET_ATTR_VAL objid:(nResourceID1) attrname:("Position") val:(
    nPositionResVal + " iuv:0")
165
166     # Create a relation that connects both elements
167     CC "Core" CREATE_CONNECTOR modelid:(nCreatedModelID) fromobjid:(
    nResourceID1) toobjid:(nTaskID1) classid:(nDecompID)
```

```
168         SETL nRelationCreated: (objid)
169         CC "Core" SET_ATTR_VAL objid:(nRelationCreated) attrname:("Positions")
        val:"EDGE 0 index:1 iuv:0"
170       }
171
172       # WORKAROUND: Switch programmatically
173       CC "Modeling" SET_ACTIVE_VARIANT variant:"" quiet
174       CC "Modeling" SET_ACTIVE_VARIANT variant:(activeVariant) quiet
175     }
176     }
177 }
```